EMS Side Serial Communications Implementation Description

Version : 0.4
Date : 20th of October 2008
Authors : Fred Cooke, based on the work of myself, Stu and others.

Foreword

The purpose of this document is to aid in understanding of the source code, as a basis for discussion on the subject, and as a reference work for those implementing the handler at the other end of the data stream. This document assumes that the latest serial specification has been read and understood.

States

The EMS can be in four states.

- Send - Sending a data packet.
- Receive - Receiving a data packet.
- Listen - Listening/Waiting for a packet /Asyncronous Data Log Preperation.
- Ignore - Ignoring incoming data while processing an already received packet.

Out of boot/reset the device will be in listen mode.

Special Bytes

Three bytes are considered special and are escaped before transmission when not being used for their specific purpose :

- Start - Indicates the start of a data packet.
- Stop - Indicates the end of a data packet.
- Escape – Used to indicate the presence of an escaped special byte following it.

Main Sections

- 1 – Flow Of Control
- 2 – Interrupt Handling Semantics
- 3 – Receiving Data In Listen Mode
- 4 – Sending Data In Send Mode
- 5 – Processing Received Packets
- 6 – something
- 7 – or other

- 8 – last one

1 - Flow Of Control

In request-response mode a device that wants to control the EMS must send request packets to it. The receive ISR code captures the packet and when complete sets a flag to indicate it is ready to be processed. Control then passes to the main loop where the flag is read and cleared and code to process the packet invoked. Depending upon the payload type an appropriate function will be called to action the request. If a response (ack or data) is required, that function will populate the buffer with the required packet and initiate the send process by loading the start byte. At this point control moves to the send ISR code which will transfer the packet data out from the buffer to the serial line. When done the send code will return the state to listening ready for the next request.

In publish-subscribe mode the main loop code will periodically prepare a datalog packet, switch off listening and initiate the send process by loading the start byte. The send ISR code will then transfer the data out as it would any other packet. Any listening devices will be able to receive regular data updates without having to construct or send any request packets. As in request-response mode, once sent receive mode will be re-enabled and any devices wishing to send a request packet will have an opportunity to do so once more. This mode may require an optional fixed or bounded period to be usable as otherwise it could potentially block reception of requests for too large a percentage of the time making communication and tuning difficult. It could however be useful to have it behave like that for non-tuning sessions and pure datalogging.

2 - Interrupt Handling Semantics

When the serial ISR runs it reads the flags register. Depending upon what is found in that register it will perform certain actions. Currently error flags are counted but otherwise ignored as it is assumed that the escaping scheme and checksum logic will catch bad data. The counting of error flag occurance is primarily to alert the user to a possible issue with their communications equipment.

If the flag indicating that the receive register is full is set then the code to handle the reception of a byte of data is run as described in section *.*. If the receive code runs, the send code does not run and vice versa with receive being the default state.

Otherwise, if the send register empty flag is set, the code to handle sending of a byte will run as described in section *.*.

Every time the serial ISR runs all relevant error flags are checked and appropriate counters incremented where required.

For specifics on what exactly is done in the serial ISR code for sending and receiving, please see sections X and Y respectively.

3 - Receiving Data In Listen Mode

Reception of data is handled by the interrupt service routine for the serial channel when triggered by the receive register full flag.

Bytes received while listening are checked to see if they are the start byte. When a start byte is found, a flag is tested to see if we are already receiving a data packet, if so an error counter is incremented and in either case the receive process is reset.

Unless the flag indicating that we are currently receiving a packet is set bytes other than the start byte are ignored. When the flag is set, and the buffer is not full bytes are processesed accordingly. If the buffer is full then an error counter is incremented and the process reset.

All bytes received while the receive flag is set are checked to see if they are the escape byte, or the stop byte.

In the case of the escape byte a flag is set to say the next byte should be unescaped. On the next iteration the byte is tested to ensure it is one of the three special bytes and the unescaped version is stored and checksummed. If one is found to not be a special byte an error counter is incremented and the process is reset.

In the case of the stop byte reception is turned off, the calculated checksum is compared with the one received in the packet. If the two match the flag is set for the packet to be processed. If the checksum does not match an error counter is incremented and the process reset.

4 - Sending Data In Send Mode

Similarly to receiving, sending will be handled by the interrupt service routine when triggered by the send register empty flag.

Initially some code in the main loop will configure the interrupt and load the transmit register with the start byte. After that each time it is triggered the handler will grab a new character from the transmission buffer and deal with it. If the byte taken from the buffer is not one of the special three it will just be sent. In the case of a byte requiring escaping an escape byte will be sent instead, and the flag indicating that the next byte should be escaped before being sent will be set. On the next iteration the special byte will be converted and sent.

Note, the checksum needs to be pre calculated and stored as part of the packet to send. This is the case because transmission on different physical interfaces works differently and it provides a consistent way to do the checksum despite those differences.

5 - Processing Received Packets

Yet to be implemented.

- **always has flags**
- always has payload id
-
- if it had an ack then it will have the same ack back.
- if it had addresses then it will have addresses. incoming source address will become destination, out address will be new source
-
- length is on a per payload ID basis.

Ensure that reception is re-enabled/reset when a packet is not sent back. And not otherwise
Return out of function or break out of case blocks
Fall through for non send cases and return for send cases?

6 - Generation Of A Packet To Transmit

Yet to be implemented.

7 - Preparation Of A Data Log Packet While Idle

Yet to be implemented.