Eightolives USB Serial Protocol

Document Number: D1000
version 1.0

April 9, 2012

Table of Contents

1 Scope

This document presents the description of the eightolives' Serial Protocol. This protocol is used to define frames of data sent between a computer with USB Serial Port and an eightolives' USB supported product.

1.1 Legal

This document and the information contained herein is provided on an "AS IS" basis and the author disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

2 Applicable Documents

1. KISS Protocol Mike Chepponis, K3MC, and Phil Karn, KA9Q
   http://www.ax25.net/kiss.aspx
2. RFC 1055 A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP
   http://tools.ietf.org/html/rfc1055
3. MCP2200 Data Sheet Microchip Technology Inc.
   http://www.microchip.com

3 Overview

The eightolives Serial Protocol defines a means of defining a frame of information to be sent between a computer and an eightolives USB supported product. The protocol is a variant of the KISS protocol described by Mike Chepponis, K3MC, and Phil Karn, KA9Q in applicable document 1. That work is an extension of the SLIP protocol described in RFC 1055.

Some eightolives products use the Microchip Technology MCP2200 USB 2.0 to UART Protocol Converter with GPIO devices. These devices provide a hardware implementation of a USB Human Interface Device (HID) and a USB Communications Device Class (CDC) implementation. The HID port is used for product configuration. It's use is defined by other product documentation. The CDC port is used as a traditional, asynchronous serial communication link that is compatible with standard Virtual Comm Port drivers. The protocol described herein provides a simple method to define frames of information sent between the computer and the product.

The protocol differs from the KISS protocol by the definition of the first byte of a frame. In this protocol the first byte is an 8 bit command whose function is product dependent. Certain commands are defined or reserved to retain compatibility with defined KISS commands.

4 USB Serial Port

The USB Serial Port is used to communicate commands, status and data once the product is functional. The default serial port settings are 8 data bits, 1 stop bit, no parity, 19,200 baud, hardware handshake enabled. The maximum supported baud rate is 926,100.

The serial interface uses a protocol to define data frames and separate a command / status data stream from a communication path data stream.

4.1 USB Serial Link Protocol

4.1.1 Special characters

The special characters are listed in Table 1.

Table 1 Special Characters

Abbreviation	Description	Hex Value
FEND	Frame End	C0
FESC	Frame Escape	DB
TFEND	Transposed Frame End	DC
TFESC	Transposed Frame Escape	DD

4.1.2 Transmission Frames

A Transmission Frame consists of the set of the FEND character, the encoded payload data stream and the FEND character.

Two FEND characters in a row should not be interpreted as delimiting an empty frame.

If a FEND ever appears in the data, it is translated into the two byte sequence FESC TFEND (Frame Escape, Transposed Frame End). Likewise, if the FESC character ever appears in the user data, it is replaced with the two character sequence FESC TFESC (Frame Escape, Transposed Frame Escape).

As characters arrive at the receiver, they are appended to a buffer containing the current frame. Receiving a FEND marks the end of the current frame.

Receipt of a FESC puts the receiver into "escaped mode", causing the receiver to translate a following TFESC or TFEND back to FESC or FEND, respectively, before adding it to the receive buffer and leaving escaped mode. Receipt of any character other than TFESC or TFEND while in escaped mode is an error; no action is taken and frame assembly continues. A TFEND or TESC received while not in escaped mode is treated as an ordinary data character.

Each asynchronous data frame sent to the radio is converted back into "pure" form and queued for processing. The maximum “pure” payload size is 128 bytes per frame.

Table 2 Frame Structure

FEND CHARACTER
COMMAND BYTE
DATA
FEND CHARACTER

Figure 1 shows a flow chart of protocol reception.

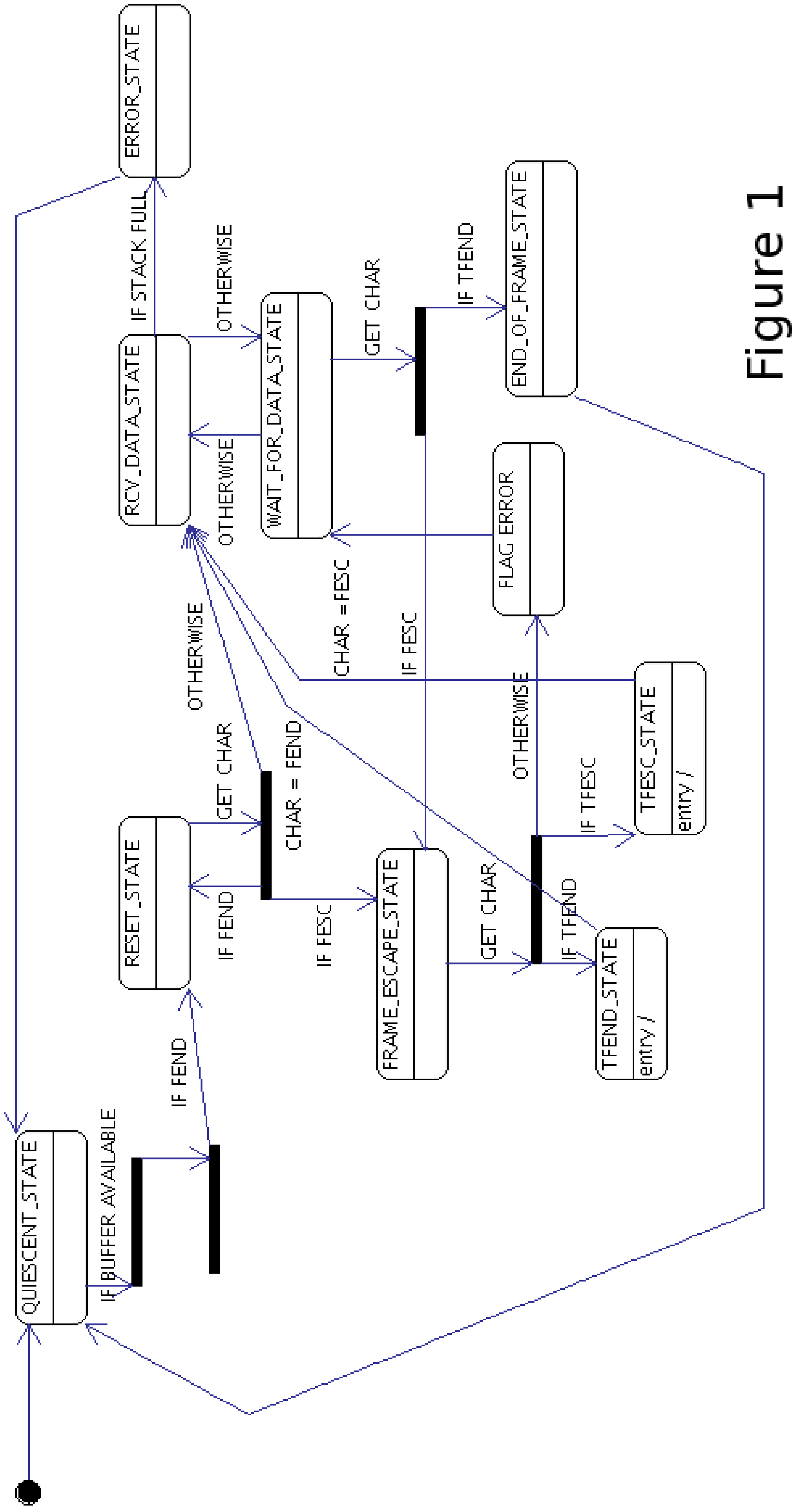
ERROR_STATE
IF STACK FULL
RCV_DATA_STATE
OTHERWISE
OTHERWISE
WAIT_FOR_DATA_STATE
GET CHAR
IF TFEND
END_OF_FRAME_STATE
FLAG ERROR
CHAR =FESC
IF FESC
OTHERWISE
OTHERWISE
TFESC_STATE
entry /
IF TFESC
GET CHAR
CHAR = FEND
RESET_STATE
IF FEND
IF FESC
FRAME_ESCAPE_STATE
GET CHAR
IF TFEND
TFEND_STATE
entry /
IF FEND
QUIESCENT_STATE
IF BUFFER AVAILABLE

Figure 1

4.1.3 Defined and Reserved Commands

The first byte of each asynchronous frame command indicator. The common command definitions are listed in Table 3. Products may define and implement additional commands.

Table 3 Basic Command Byte Definitions

Command	Function	Comments
0x00	DATA FRAME	Content is data for transmission or reception
0x01	Reserved	(TXDELAY)
0x02	Reserved	(P)
0x03	Reserved	(SlotTime)
0x04	Reserved	(TXTail)
0x05	Reserved	(Full Duplex)
0x06	Reserved	(SetHardware)
0xFF	Reserved	(Return)
0x08	GET INFO	Returns ASCII String: manufacturer, model number, revision
0x09	GET CAPABILITIES	Returns ASCII String listing capabilities, limits and options
0x0A	READ REGISTER	Byte 2 specifies register address
0x0B	WRITE REGISTER	Byte 2 specifies register address, Byte 3 specifies data to write
0x0C	INTERRUPT	Byte 2 is an identifying number. This is normally sent by the product to alert the computer.

4.1.4 Response Frames

Frames sent in response to a command have the command byte set to the inverse of the command.

5 Product Implementations

5.1 Extensions

Products implementing this protocol may extend the command set. Product documentation will define all additional commands, responses and data formats supported by the product.