CRSF Protocol

14.08.2017 Rev 07

Table of content

Features

- Low latency high update rate for RC signals between RC - XF and XF - FC
- Bidirectional communication
- Share telemetry from flying platform to the RC
- Edit configuration for direct connected devices and remotely connected devices (RC can configure FC or OSD over crossfire)
- Share receiver serial number to TX so it can be matched to model memory.

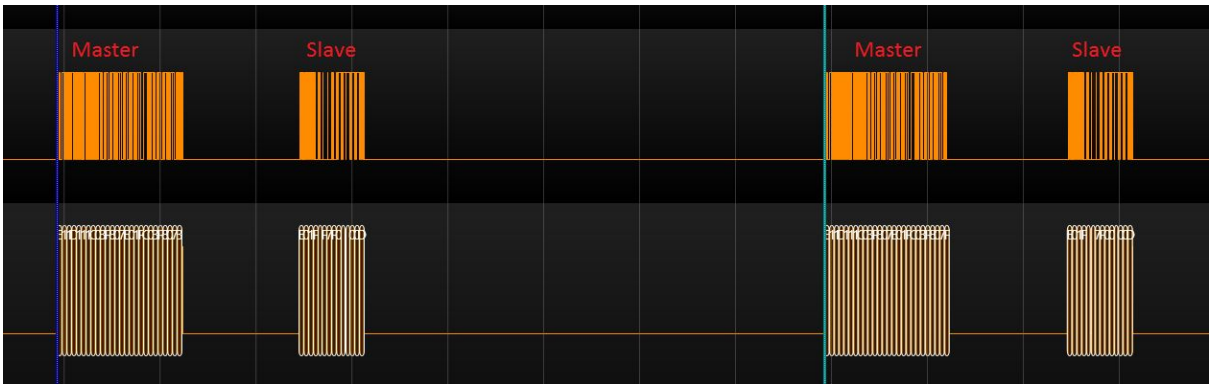
Future features:

- System should be able to handle multiple of the same sensors using sensor ID’s

Hardware

Single wire half duplex UART

This configuration is usually used between RC and Crossfire TX. XF TX support both inverted and non-inverted UART. The RC acts as master in this case and the XF TX can only send telemetry if it's synchronized to the RC frames sent by the RC. The RC frame update rate should not be sent more often than every 4ms. The UART runs at 400kbaud 8N1 (inverted or non-inverted) at 3.3V level.



Dual wire / full duplex UART

This configuration is usually used on the flying platform side. Two devices are connected by regular UART connection. Only non-inverted (regular) UART is supported in this configuration. The UART runs at 400kbaud 8N1 at 3.0 to 3.3V level.

Multimaster I2C (BST)

BST is a multimaster I2C bus. It runs at 3.3V level at 100kHz using 7 bit addresses. Device addresses already contain the R/W bit. Which means the list is each device's write address and read address is Device addresses + 1.
Each device supporting BST should release SDA in any case to not block the bus. It's recommended to monitor the heartbeat message and reset the interface if there is a timeout >1.5s. It's required to support general call frames which will be called broadcast frames within this document.

Frame

Structure

The basic structure for each frame is the same. There is a range of Types with an extended header which will have the first few bytes of payload standardized. This is required to route frame across multiple devices for point to point communication.

Broadcast Frames:
<Device address or Sync Byte> <Frame length> <Type><Payload> <CRC>

Extended header frames:
<Device address or Sync Byte> <Frame length> <Type><Destination Address> <Origin Address> <Payload> <CRC>

Device address or Sync Byte:	(uint8_t) Device address for I2C or Sync byte serial connections.In case of I2C (BST) this is mostly “Broadcast address” or defined by Router.
Frame length:	Amount of bytes including Type, Payload and CRC (uint8_t)
Type:	Frame type (uint8_t)
CRC:	8 Bit CRC of the frame. See CRC (uint8_t)
Sync Byte:	0xC8
Endianness	Big endian

Device addresses

0x00	Broadcast address
0x10	USB Device
0x12	Bluetooth Module
0x80	TBS CORE PNP PRO
0x8A	Reserved
0xC0	PNP PRO digital current sensor
0xC2	PNP PRO GPS
0xC4	TBS Blackbox
0xC8	Flight controller
0xCA	Reserved
0xCC	Race tag
0xEA	Radio Transmitter
0xEB	Reserved
0xEC	Crossfire / UHF receiver
0xEE	Crossfire transmitter

CRC

CRC includes Type and Payload of each frame.

Code example:

/* CRC8 implementation with polynom = $x^7 + x^6 + x^4 + x^2 + x^0$ (0xD5) */

```
unsigned char crc8tab[256] = {
0x00, 0xD5, 0x7F, 0xAA, 0xFE, 0x2B, 0x81, 0x54, 0x29, 0xFC, 0x56, 0x83, 0xD7, 0x02, 0xA8, 0x7D,
0x52, 0x87, 0x2D, 0xF8, 0xAC, 0x79, 0xD3, 0x06, 0x7B, 0xAE, 0x04, 0xD1, 0x85, 0x50, 0xFA, 0x2F,
0xA4, 0x71, 0xDB, 0x0E, 0x5A, 0x8F, 0x25, 0xF0, 0x8D, 0x58, 0xF2, 0x27, 0x73, 0xA6, 0x0C, 0xD9,
0xF6, 0x23, 0x89, 0x5C, 0x08, 0xDD, 0x77, 0xA2, 0xDF, 0x0A, 0xA0, 0x75, 0x21, 0xF4, 0x5E, 0x8B,
0x9D, 0x48, 0xE2, 0x37, 0x63, 0xB6, 0x1C, 0xC9, 0xB4, 0x61, 0xCB, 0x1E, 0x4A, 0x9F, 0x35, 0xE0,
0xCF, 0x1A, 0xB0, 0x65, 0x31, 0xE4, 0x4E, 0x9B, 0xE6, 0x33, 0x99, 0x4C, 0x18, 0xCD, 0x67, 0xB2,
0x39, 0xEC, 0x46, 0x93, 0xC7, 0x12, 0xB8, 0x6D, 0x10, 0xC5, 0x6F, 0xBA, 0xEE, 0x3B, 0x91, 0x44,
0x6B, 0xBE, 0x14, 0xC1, 0x95, 0x40, 0xEA, 0x3F, 0x42, 0x97, 0x3D, 0xE8, 0xBC, 0x69, 0xC3, 0x16,
0xEF, 0x3A, 0x90, 0x45, 0x11, 0xC4, 0x6E, 0xBB, 0xC6, 0x13, 0xB9, 0x6C, 0x38, 0xED, 0x47, 0x92,
0xBD, 0x68, 0xC2, 0x17, 0x43, 0x96, 0x3C, 0xE9, 0x94, 0x41, 0xEB, 0x3E, 0x6A, 0xBF, 0x15, 0xC0,
0x4B, 0x9E, 0x34, 0xE1, 0xB5, 0x60, 0xCA, 0x1F, 0x62, 0xB7, 0x1D, 0xC8, 0x9C, 0x49, 0xE3, 0x36,
0x19, 0xCC, 0x66, 0xB3, 0xE7, 0x32, 0x98, 0x4D, 0x30, 0xE5, 0x4F, 0x9A, 0xCE, 0x1B, 0xB1, 0x64,
0x72, 0xA7, 0x0D, 0xD8, 0x8C, 0x59, 0xF3, 0x26, 0x5B, 0x8E, 0x24, 0xF1, 0xA5, 0x70, 0xDA, 0x0F,
0x20, 0xF5, 0x5F, 0x8A, 0xDE, 0x0B, 0xA1, 0x74, 0x09, 0xDC, 0x76, 0xA3, 0xF7, 0x22, 0x88, 0x5D,
0xD6, 0x03, 0xA9, 0x7C, 0x28, 0xFD, 0x57, 0x82, 0xFF, 0x2A, 0x80, 0x55, 0x01, 0xD4, 0x7E, 0xAB,
0x84, 0x51, 0xFB, 0x2E, 0x7A, 0xAF, 0x05, 0xD0, 0xAD, 0x78, 0xD2, 0x07, 0x53, 0x86, 0x2C, 0xF9};

uint8_t crc8(const uint8_t * ptr, uint8_t len)
{
  uint8_t crc = 0;
  for (uint8_t i=0; i<len; i++) {
    crc = crc8tab[crc ^ *ptr++];
  }
  return crc;
}
```

Routing

If a device has more than one CRSF port it's required to forward all received frames to the other ports. CRSF works as a star network. It's forbidden to use any loop connection as it would keep forwarding the same message endlessly. This can be detected if one device receives it's own Parameter ping devices request or Heartbeat message it should file an error and tell the user.

Frames within Broadcast frames range will be forwarded to any other CRSF port.

Received frames within Extended header frames range should be parsed. The origin address should be stored in a lookup table. Each other port's lookup table should then be searched for matching the destination address. In case of a found match the frame should only be forwarded to the matching port. If there is no match the frame will be forwarded to any other CRSF port except the one the frame was received.

Frame Types

The following list shows the content of each frame type. The most common frames have an own type. Commands and others share one type.

If a value is unknown, variable max value will be sent instead.

Broadcast frames

Type range: 0x00 to 0x27

0x02 GPS

Payload:

- int32_t Latitude (degree / 10`000`000)
- int32_t Longitude (degree / 10`000`000)
- uint16_t Groundspeed (km/h / 100)
- uint16_t GPS heading (degree / 100)
- uint16 Altitude (meter - 1000m offset)
- uint8_t Satellites in use (counter)

0x08 Battery sensor

Payload:

- uint16_t Voltage (mV * 100)
- uint16_t Current (mA * 100)
- uint24_t Capacity (mAh)
- uint8_t Battery remaining (percent)

0x0B Heartbeat

Payload:

- uint8_t Origin Device address

0x0F Video transmitter

VTX Frequency table:

```
{ 5865, 5845, 5825, 5805, 5785, 5765, 5745, 5725},    /* Band A */
{ 5733, 5752, 5771, 5790, 5809, 5828, 5847, 5866},    /* Band B */
{ 5705, 5685, 5665, 5645, 5885, 5905, 5925, 5945},    /* Band E */
{ 5740, 5760, 5780, 5800, 5820, 5840, 5860, 5880},    /* Ariwave */
{ 5658, 5695, 5732, 5769, 5806, 5843, 5880, 5917},    /* Race */
{ 5621, 5584, 5547, 5510, 5473, 5436, 5399, 5362}};   /* LO Race */
```

Payload:

- uint8_t Origin address
- uint8_t Status (bit7-5 enum SmartAudio_V1 = 0, SmartAudio_V2 /
  bit4 bool VTX_is_available /
  bit1 bool is_in_user_frequency_mode /
  bit0 bool is_in_PitMode)
- uint8_t Band_Channel (see frequency table)
- uint16_t User_Frequency
- uint8_t PitMode_and_Power
  (bit7-4 enum off = 0, In_Band, Out_Band /
  bit3-0 enum 25mW = 0, 200mW, 500mW 800mW)

0x14 Link statistics

Payload:

- uint8_t Uplink RSSI Ant. 1 (dBm * -1)
- uint8_t Uplink RSSI Ant. 2 (dBm * -1)
- uint8_t Uplink Package success rate / Link quality (%)
- int8_t Uplink SNR (db)
- uint8_t Diversity active antenna (enum ant. 1 = 0, ant. 2)
- uint8_t RF Mode (enum 4fps = 0 , 50fps, 150hz)

- uint8_t Uplink TX Power (enum 0mW = 0, 10mW, 25 mW, 100 mW, 500 mW, 1000 mW, 2000mW)
- uint8_t Downlink RSSI (dBm * -1)
- uint8_t Downlink package success rate / Link quality (%)
- int8_t Downlink SNR (db)

Uplink is the connection from the ground to the UAV and downlink the opposite direction.

0x16 RC channels packed

Payload:

- 11bits Channel 1
- 11bits Channel 2
- …
- 11bits Channel 16

16 channels packed into 22 bytes.
Center (1500us) = 992
TICKS_TO_US(x) ((x - 992) * 5 / 8 + 1500)
US_TO_TICKS(x) ((x - 1500) * 8 / 5 + 992)

0x1E Attitude

Payload:

- int16_t Pitch angle (rad / 10000)
- int16_t Roll angle (rad / 10000)
- int16_t Yaw angle (rad / 10000)

0x21 Flight mode text based

Payload:

- char[] Flight mode (Null-terminated string)

Extended header frames

Type range: 0x28 to 0x96

0x28 Parameter ping devices

The host can ping a specific device (destination node address of device) or all devices (destination node address 0x00 Broadcast address) and they will answer with a Parameter device information frame

Payload:

- uint8_t Destination node address
- uint8_t Origin node address

0x29 Parameter device information

Payload:

- uint8_t Destination node address
- uint8_t Device node address
- char[] Device name (Null-terminated string)
- uint32_t Serial number
- uint32_t Hardware ID
- uint32_t Firmware ID
- uint8_t Parameters count
- uint8_t Parameter version number

0x2B Parameter settings (entry)

This is how a device (node address) can share a parameter to another device.
Payload:

- uint8_t Destination node address
- uint8_t Origin node address
- uint8_t Parameter number (starting from 1)
- uint8_t Parameter chunks remaining (Chunks)
- uint8_t Parent folder (parameter number of the parent folder, 0 means root of the device)
- enum data_type Data type (Parameter type definitions and hidden bit)
- char[] Name (Null-terminated string)
- Value (size depending on data type)
- Min value (size depending on data type)
- Max value (size depending on data type)
- Default value (size depending on data type)
- uint8_t Decimal point (type float only otherwise this entry is not sent)
- int32_t Step size (type float only otherwise this entry is not sent)
- char[] Unit (Null-terminated string / not sent for type string and folder)
- uint8_t String max length (for string type only)

Parent parameter: Links to a parent folder for better menu structure

0x2C Parameter settings (read)

Request a specific parameter. This command is for re-request parameters if the didn't make it through the RF link.
Payload:

- uint8_t Destination node address
- uint8_t Origin node address
- uint8_t Parameter number
- uint8_t Parameter chunk number (Chunks)

Timeout:

- 2s Failed timeout

0x2D Parameter value (write)

This command is for override a parameter. The destination node will answer with a Parameter value frame sent to the origin node address for verification.
Payload:

- uint8_t Destination node address
- uint8_t Origin node address
- uint8_t Parameter number
- Data (size depending on data type)

Timeout:

- 2s Failed timeout

Chunks

Maximum frame size is 62 bytes (64 - header).

The host should always read (0x2C Parameter settings (read)) chunk number 0 by default. If the read parameter (0x2B Parameter settings (entry)) fits the maximum size it will answer with chunks remaining 0 inside the parameter frame. Otherwise it will send how many chunks are left to read.

Example:

- Host: Read Param 2, Chunk 0
- Device: Sends Parameter 2, Chunk 2 (parameter is too big to fit 62 bytes. The device split it up in 3 chunks. First chunk is sent with this frame and 2 are remaining)
- Host: Read Param 2, Chunk 1 (as the device answered with a chunk size >0 the host keeps reading the same parameter and increases the chunk count)
- Device: Sends Parameter 2, Chunk 1
- Host: Read Param 2, Chunk 2
- Device: Sends Parameter 2, Chunk 0
- Host: Read Param 3, Chunk 0 (Host received last chunk so it can ready any other parameter starting over with chunk 0

Parameter type definitions and hidden bit

Parameter type is 8bit wide. The bit 7 indicates if the parameter is hidden (1 = hidden / 0 = visible). This gives the ability to dynamically show or hide parameters depending on other parameters. Bit 6-0 holds the type of parameter information (enum data_type).

```
enum data_type
{
        UINT8 = 0
        INT8 = 1
        UINT16 = 2
        INT16 = 3
        FLOAT  = 8
        TEXT_SELECTION = 9
        STRING = 10
        FOLDER = 11
        INFO = 12
        COMMAND = 13
        OUT_OF_RANGE = 127
}
```

FLOAT:
Value, min, max and default are sent as a INT32. The decimal point value tells how many digits of the value are behind the decimal point. Step size is the recommended increment or decrement value to modify the value.

TEXT_SELECTION:
The value part of this entry is separated in two parts. First part is a char array with all possible values in text format. They are separated by a semicolon (;) and the array is null-terminated at the end. The second part is a uint8_t variable with the current value. The min, max and default value is represented as uint8_t number where a 0 represent the first text. To modify this parameter only the uint8_t value needs to be sent for the new value.

STRING:
This type is for text modification. Only the current text and the default text will be transmitted. There is no min and max entry sent for this type.

FOLDER:
Folder is used to make a better structure of the parameters. Every parameter has a parent entry where the parameter can link to the parent folder.

INFO:
Value is a null terminated string. This entry can’t be modified.

COMMAND:
With type command the host is able to run/execute function on a device. This can be anything link bind crossfire, calibrate gyro/acc ect.
The device default state is ready. Once the host want to execute the function it writes the parameter with status START. Depending on the function the device switches to PROGRESS, CONFIRMATION_NEEDED or READY. When the device sends CONFIRMATION_NEEDED the host will show a confirmation box with “confirm” or “cancel” selection. If the use select one the selection will be transmitted to the device and the function continues to execute.
With the entry info the device can send additional information the host. If the host sends status POLL, it will force the device to send an updated status of the 0x2B Parameter settings (entry).

Example:

- Host read parameter
- Device send parameter: COMMAND, Name = Bind, Status = READY, Info = NULL
- Host write: Status = START
- Device send parameter: COMMAND, Name = Bind, Status = PROGRESS, Info = Binding
- Host can write: Status = POLL (optional)
- Host can write: Status = POLL (optional)
- Host can write: Status = POLL (optional)
- Device completed bind process
- Device send parameter: COMMAND, Name = Bind, Status = READY, Info = OK

0x2B Parameter settings (entry) in case of type COMMAND:

- uint8_t Destination node address
- uint8_t Origin node address
- uint8_t Parameter number
- uint8_t Parent folder
- enum data_type Data type (COMMAND)
- char[] Name (Null-terminated string)
- uint8_t Status
- uint8_t Timeout (ms * 100)
- char[] Info (Null-terminated string)

```
enum cmd_status
{
        READY = 0,
        START = 1,
        PROGRESS = 2,
        CONFIRMATION_NEEDED = 3,
        CONFIRM = 4,
        CANCEL = 5,
        POLL = 6
}
```

OUT_OF_RANGE:
This type will be sent if a parameter number out of the device parameter range will be requested. It will be also sent as last parameter to let the host know the end of parameter list on a Parameters settings list (read request).

Commands

0x32 Command frame

Payload:

- uint8_t Destination node address
- uint8_t Origin node address
- uint8_t Command ID
- uint8_t [] Payload depending on Command ID
- uint8_t Command_CRC8 (8 bit CRC POLYNOM = 0xBA)

Command_CRC8 is for (0x32 Command frame) check up only, it is part of the Commands Payload, and it includes Command frame, Destination, Origin, Command ID and Payload of each Command Frame.
Note: You will also need to include CRC at the end for the full Frame

Code example:

/* Command_CRC8 implementation with polynom = $x^7 + x^5 + x^4 + x^3 + x^1$ (0xBA) */
unsigned char command_crc8tab[256] = {
0x00, 0xBA, 0xCE, 0x74, 0x26, 0x9C, 0xE8, 0x52, 0x4C, 0xF6, 0x82, 0x38, 0x6A, 0xD0, 0xA4, 0x1E,
0x98, 0x22, 0x56, 0xEC, 0xBE, 0x04, 0x70, 0xCA, 0xD4, 0x6E, 0x1A, 0xA0, 0xF2, 0x48, 0x3C, 0x86,
0x8A, 0x30, 0x44, 0xFE, 0xAC, 0x16, 0x62, 0xD8, 0xC6, 0x7C, 0x08, 0xB2, 0xE0, 0x5A, 0x2E, 0x94,
0x12, 0xA8, 0xDC, 0x66, 0x34, 0x8E, 0xFA, 0x40, 0x5E, 0xE4, 0x90, 0x2A, 0x78, 0xC2, 0xB6, 0x0C,
0xAE, 0x14, 0x60, 0xDA, 0x88, 0x32, 0x46, 0xFC, 0xE2, 0x58, 0x2C, 0x96, 0xC4, 0x7E, 0x0A, 0xB0,
0x36, 0x8C, 0xF8, 0x42, 0x10, 0xAA, 0xDE, 0x64, 0x7A, 0xC0, 0xB4, 0x0E, 0x5C, 0xE6, 0x92, 0x28,
0x24, 0x9E, 0xEA, 0x50, 0x02, 0xB8, 0xCC, 0x76, 0x68, 0xD2, 0xA6, 0x1C, 0x4E, 0xF4, 0x80, 0x3A,
0xBC, 0x06, 0x72, 0xC8, 0x9A, 0x20, 0x54, 0xEE, 0xF0, 0x4A, 0x3E, 0x84, 0xD6, 0x6C, 0x18, 0xA2,
0xE6, 0x5C, 0x28, 0x92, 0xC0, 0x7A, 0x0E, 0xB4, 0xAA, 0x10, 0x64, 0xDE, 0x8C, 0x36, 0x42, 0xF8,
0x7E, 0xC4, 0xB0, 0x0A, 0x58, 0xE2, 0x96, 0x2C, 0x32, 0x88, 0xFC, 0x46, 0x14, 0xAE, 0xDA, 0x60,
0x6C, 0xD6, 0xA2, 0x18, 0x4A, 0xF0, 0x84, 0x3E, 0x20, 0x9A, 0xEE, 0x54, 0x06, 0xBC, 0xC8, 0x72,
0xF4, 0x4E, 0x3A, 0x80, 0xD2, 0x68, 0x1C, 0xA6, 0xB8, 0x02, 0x76, 0xCC, 0x9E, 0x24, 0x50, 0xEA,
0x48, 0xF2, 0x86, 0x3C, 0x6E, 0xD4, 0xA0, 0x1A, 0x04, 0xBE, 0xCA, 0x70, 0x22, 0x98, 0xEC, 0x56,
0xD0, 0x6A, 0x1E, 0xA4, 0xF6, 0x4C, 0x38, 0x82, 0x9C, 0x26, 0x52, 0xE8, 0xBA, 0x00, 0x74, 0xCE,
0xC2, 0x78, 0x0C, 0xB6, 0xE4, 0x5E, 0x2A, 0x90, 0x8E, 0x34, 0x40, 0xFA, 0xA8, 0x12, 0x66, 0xDC,
0x5A, 0xE0, 0x94, 0x2E, 0x7C, 0xC6, 0xB2, 0x08, 0x16, 0xAC, 0xD8, 0x62, 0x30, 0x8A, 0xFE, 0x44} ;

You may use the same CRC code in above.

Feature or device related command IDs:

- 0x01 FC Commands:
    - 0x01 Force Disarm:
    - 0x02 Scale Channel

- 0x03 Bluetooth Command
    - 0x01 Reset
    - 0x02 Enable
        - Uint8_t Enable (0 = disable, 1 = enable)
    - 0x64 Echo

- 0x05 OSD Commands:
    - 0x01 Send Buttons:
        - uint8_t Buttons bitwise (Bit 7=Enter, 6=Up, 5=Down, 4=Left, 3=Right)

- 0x08 VTX Commands:
    - 0x01 Change channel
        - uint8_t Channel Number (0-47)
    - 0x02 Change frequency
        - uint16_t Frequency (5000-6000MHz)
    - 0x03 Change power
        - uint8_t Power enum (0 = 25mW, 1 = 200mW, 2= 500mW, 3 = 800mW)
    - 0x04 Change PitMode
        - uint8_t PitMode enum (0 = OFF, 1 = In_Band, 2 = Out_Band)
    - 0x05 Power up from PitMode (bare command)

- 0x09 LED
    - 0x01 Set to default (revert to target specific settings)
    - 0x02 Override LED color (packed)
        - 9 bits H (0-359°)
        - 7 bits S (0-100%)
        - 8 bits V (0-100%)
    - 0x03 Override pulse (packed)
        - uint16 duration (milliseconds from start color to stop color)
        - 9 bits H_Start (0-359°)
        - 7 bits S_Start (0-100%)
        - 8 bits V_Start (0-100%)
        - 9 bits H_Stop (0-359°)
        - 7 bits S_Stop (0-100%)
        - 8 bits V_Stop (0-100%)
    - 0x03 Override blink (packed)
        - uint16 Intervall
        - 9 bits H_Start (0-359°)
        - 7 bits S_Start (0-100%)
        - 8 bits V_Start (0-100%)
        - 9 bits H_Stop (0-359°)
        - 7 bits S_Stop (0-100%)
        - 8 bits V_Stop (0-100%)
    - 0x04 Override shift (packed)
        - uint16 Intervall
        - 9 bits H (0-359°)
        - 7 bits S (0-100%)
        - 8 bits V (0-100%)

General Command IDs:

- 0x0A Firmware Update:
    - 0x0A Start Bootloader (bare command)
    - 0x0B Erase memory (bare command)

0x78 - 0x79 KISS FC

Reserved type range for KISS FC

Changelog

Rev 07

- Use packed frames for LED commands

Rev 06

- Direct commands added
- Command CRC8
- KISS FC Typ range added
- LED override command added

Rev 05

- Routing added
- Better separation for broadcast and extended header frames added
- I2C hardware documentation added

Rev 04

- Example formula for RC timing calculation

Rev 03

- Parameters added

Rev 02

- First version