/\*

CRSF frame has the structure:

<Device address> <Frame length> <Type> <Payload> <CRC>

Device address: (uint8\_t)

Frame length: length in bytes including Type (uint8\_t)

Type: (uint8\_t)

Payload: various types

CRC: (uint8\_t), crc of <Type> and <Payload>

\*/

/\*

0x02 GPS

Payload:

int32\_t Latitude ( degree / 10`000`000 )

int32\_t Longitude (degree / 10`000`000 )

uint16\_t Groundspeed ( km/h / 10 )

uint16\_t GPS heading ( degree / 100 )

uint16 Altitude ( meter ­1000m offset )

uint8\_t Satellites in use ( counter )

\*/

/\*

0x02 GPS

Payload:

int32\_t Latitude ( degree / 10`000`000 )

int32\_t Longitude (degree / 10`000`000 )

uint16\_t Groundspeed ( km/h / 10 )

uint16\_t GPS heading ( degree / 100 )

uint16 Altitude ( meter ­1000m offset )

uint8\_t Satellites in use ( counter )

\*/

/\*

0x07 Vario sensor

Payload:

int16 Vertical speed ( cm/s )

\*/

/\*

0x08 Battery sensor

Payload:

uint16\_t Voltage ( mV \* 100 )

uint16\_t Current ( mA \* 100 )

uint24\_t Capacity ( mAh )

uint8\_t Battery remaining ( percent )

\*/

/\*

0x1E Attitude

Payload:

int16\_t Pitch angle ( rad / 10000 )

int16\_t Roll angle ( rad / 10000 )

int16\_t Yaw angle ( rad / 10000 )

\*/

static void crsfFrameFlightMode(sbuf\_t \*dst)

{

// just do "OK" for the moment as a placeholder

// write zero for frame length, since we don't know it yet

uint8\_t \*lengthPtr = sbufPtr(dst);

sbufWriteU8(dst, 0);

crsfSerialize8(dst, CRSF\_FRAMETYPE\_FLIGHT\_MODE);

// use same logic as OSD, so telemetry displays same flight text as OSD when armed

const char \*flightMode = "OK";

if (ARMING\_FLAG(ARMED)) {

if (STATE(AIRMODE\_ACTIVE)) {

flightMode = "AIR";

} else {

flightMode = "ACRO";

}

if (FLIGHT\_MODE(FAILSAFE\_MODE)) {

flightMode = "!FS!";

} else if (IS\_RC\_MODE\_ACTIVE(BOXHOMERESET) && !FLIGHT\_MODE(NAV\_RTH\_MODE) && !FLIGHT\_MODE(NAV\_WP\_MODE)) {

flightMode = "HRST";

} else if (FLIGHT\_MODE(MANUAL\_MODE)) {

flightMode = "MANU";

} else if (FLIGHT\_MODE(NAV\_RTH\_MODE)) {

flightMode = "RTH";

} else if (FLIGHT\_MODE(NAV\_POSHOLD\_MODE)) {

flightMode = "HOLD";

} else if (FLIGHT\_MODE(NAV\_COURSE\_HOLD\_MODE) && FLIGHT\_MODE(NAV\_ALTHOLD\_MODE)) {

flightMode = "CRUZ";

} else if (FLIGHT\_MODE(NAV\_COURSE\_HOLD\_MODE)) {

flightMode = "CRSH";

} else if (FLIGHT\_MODE(NAV\_WP\_MODE)) {

flightMode = "WP";

} else if (FLIGHT\_MODE(NAV\_ALTHOLD\_MODE)) {

flightMode = "AH";

} else if (FLIGHT\_MODE(ANGLE\_MODE)) {

flightMode = "ANGL";

} else if (FLIGHT\_MODE(HORIZON\_MODE)) {

flightMode = "HOR";

} else if (FLIGHT\_MODE(ANGLEHOLD\_MODE)) {

flightMode = "ANGH";

}

#ifdef USE\_GPS

} else if (feature(FEATURE\_GPS) && navConfig()->general.flags.extra\_arming\_safety && (!STATE(GPS\_FIX) || !STATE(GPS\_FIX\_HOME))) {

flightMode = "WAIT"; // Waiting for GPS lock

#endif

} else if (isArmingDisabled()) {

flightMode = "!ERR";

}

/\*

0x29 Device Info

Payload:

uint8\_t Destination

uint8\_t Origin

char[] Device Name ( Null terminated string )

uint32\_t Null Bytes

uint32\_t Null Bytes

uint32\_t Null Bytes

uint8\_t 255 (Max MSP Parameter)

uint8\_t 0x01 (Parameter version 1)

\*/

<https://github.com/AlfredoSystems/ArduinoCRSF>

**DEST - Destination address or "sync" byte**

* CRSF\_ADDRESS\_CRSF\_TRANSMITTER = (0xEE) //Going to the transmitter module
* CRSF\_ADDRESS\_RADIO\_TRANSMITTER = (0xEA) //Going to the handset
* CRSF\_ADDRESS\_FLIGHT\_CONTROLLER = (0xC8) //Going to the flight controller
* CRSF\_ADDRESS\_CRSF\_RECEIVER = (0xEC) //Going to the receiver (from FC)

**LEN - Length of bytes that follow**

Overall packet length is PayloadLength+4 (dest, len, type, crc), or LEN+2 (dest, len).

**TYPE - CRSF\_FRAMETYPE**

* CRSF\_FRAMETYPE\_GPS = 0x02,
* CRSF\_FRAMETYPE\_VARIO = 0x07,
* CRSF\_FRAMETYPE\_BATTERY\_SENSOR = 0x08,
* CRSF\_FRAMETYPE\_BARO\_ALTITUDE = 0x09,
* CRSF\_FRAMETYPE\_LINK\_STATISTICS = 0x14,
* CRSF\_FRAMETYPE\_OPENTX\_SYNC = 0x10,
* CRSF\_FRAMETYPE\_RADIO\_ID = 0x3A,
* CRSF\_FRAMETYPE\_RC\_CHANNELS\_PACKED = 0x16,
* CRSF\_FRAMETYPE\_LINK\_RX\_ID = 0x1C,
* CRSF\_FRAMETYPE\_LINK\_TX\_ID = 0x1D,
* CRSF\_FRAMETYPE\_ATTITUDE = 0x1E,
* CRSF\_FRAMETYPE\_FLIGHT\_MODE = 0x21, // Extended Header Frames, range: 0x28 to 0x96
* CRSF\_FRAMETYPE\_DEVICE\_PING = 0x28,
* CRSF\_FRAMETYPE\_DEVICE\_INFO = 0x29,
* CRSF\_FRAMETYPE\_PARAMETER\_SETTINGS\_ENTRY = 0x2B,
* CRSF\_FRAMETYPE\_PARAMETER\_READ = 0x2C,
* CRSF\_FRAMETYPE\_PARAMETER\_WRITE = 0x2D,
* CRSF\_FRAMETYPE\_COMMAND = 0x32, // KISS frames
* CRSF\_FRAMETYPE\_KISS\_REQ = 0x78,
* CRSF\_FRAMETYPE\_KISS\_RESP = 0x79, // MSP commands
* CRSF\_FRAMETYPE\_MSP\_REQ = 0x7A,
* CRSF\_FRAMETYPE\_MSP\_RESP = 0x7B,
* CRSF\_FRAMETYPE\_MSP\_WRITE = 0x7C, // Ardupilot frames
* CRSF\_FRAMETYPE\_ARDUPILOT\_RESP = 0x80,

**CRC - CRC8 using poly 0xD5**

Includes all bytes from type (buffer[2]) to end of payload.

**Payload of each frametype**

**CRSF\_FRAMETYPE\_GPS = 0x02**

* int32\_t latitude; // degree / 10,000,000 big endian
* int32\_t longitude; // degree / 10,000,000 big endian
* uint16\_t groundspeed; // km/h / 10 big endian
* uint16\_t heading; // GPS heading, degree/100 big endian
* uint16\_t altitude; // meters, +1000m big endian
* uint8\_t satellites; // satellites

**CRSF\_FRAMETYPE\_VARIO = 0x07**

* int16\_t verticalspd; // Vertical speed in cm/s, BigEndian

**CRSF\_FRAMETYPE\_BATTERY\_SENSOR = 0x08**

* unsigned voltage : 16; // V \* 10 big endian
* unsigned current : 16; // A \* 10 big endian
* unsigned capacity : 24; // mah big endian
* unsigned remaining : 8; // %

**CRSF\_FRAMETYPE\_BARO\_ALTITUDE = 0x09**

* uint16\_t altitude; // Altitude in decimeters + 10000dm, or Altitude in meters if high bit is set, BigEndian
* int16\_t verticalspd; // Vertical speed in cm/s, BigEndian

**CRSF\_FRAMETYPE\_HEARTBEAT = 0x0B**

* uint8\_t Origin Device address;

**CRSF\_FRAMETYPE\_VIDEO\_TRANSMITTER = 0x0F**

* uint8\_t Origin address;
* uint8\_t Status;
* uint8\_t Band\_Channel;
* uint16\_t User\_Frequency;
* uint8\_t PitMode\_and\_Power;

**CRSF\_FRAMETYPE\_LINK\_STATISTICS = 0x14**

* uint8\_t uplink\_RSSI\_1;
* uint8\_t uplink\_RSSI\_2;
* uint8\_t uplink\_Link\_quality;
* int8\_t uplink\_SNR;
* uint8\_t active\_antenna;
* uint8\_t rf\_Mode;
* uint8\_t uplink\_TX\_Power;
* uint8\_t downlink\_RSSI;
* uint8\_t downlink\_Link\_quality;
* int8\_t downlink\_SNR;

**CRSF\_FRAMETYPE\_OPENTX\_SYNC = 0x10**

* ????

**CRSF\_FRAMETYPE\_RADIO\_ID = 0x3A**

* uint16\_t radioAddress; //should be 0xEA00?
* uint8\_t timingCorrectionFrame; //should be 0x10?
* uint32\_t update\_interval; //what is this?
* int32\_t offset; //what is this?

**CRSF\_FRAMETYPE\_RC\_CHANNELS\_PACKED = 0x16**

* unsigned ch0 : 11;
* unsigned ch1 : 11;
* unsigned ch2 : 11;
* unsigned ch3 : 11;
* unsigned ch4 : 11;
* unsigned ch5 : 11;
* unsigned ch6 : 11;
* unsigned ch7 : 11;
* unsigned ch8 : 11;
* unsigned ch9 : 11;
* unsigned ch10 : 11;
* unsigned ch11 : 11;
* unsigned ch12 : 11;
* unsigned ch13 : 11;
* unsigned ch14 : 11;
* unsigned ch15 : 11;

**CRSF\_FRAMETYPE\_LINK\_RX\_ID = 0x1C**

* uint8\_t rxRssiPercent;
* uint8\_t rxRfPower; //should be signed int?

**CRSF\_FRAMETYPE\_LINK\_TX\_ID = 0x1D**

* uint8\_t txRssiPercent;
* uint8\_t txRfPower; //should be signed int?
* uint8\_t txFps;

**CRSF\_FRAMETYPE\_ATTITUDE = 0x1E**

* uint16\_t pitch; // pitch in radians, BigEndian
* uint16\_t roll; // roll in radians, BigEndian
* uint16\_t yaw; // yaw in radians, BigEndian

**CRSF\_FRAMETYPE\_FLIGHT\_MODE = 0x21**

* char[]; //Flight mode ( Null-terminated string ) // Extended Header Frames, range: 0x28 to 0x96

**CRSF\_FRAMETYPE\_DEVICE\_PING = 0x28,**

* ????

**CRSF\_FRAMETYPE\_DEVICE\_INFO = 0x29,**

* ????

**CRSF\_FRAMETYPE\_PARAMETER\_SETTINGS\_ENTRY = 0x2B,**

* ????

**CRSF\_FRAMETYPE\_PARAMETER\_READ = 0x2C,**

* ????

**CRSF\_FRAMETYPE\_PARAMETER\_WRITE = 0x2D,**

* ????

**CRSF\_FRAMETYPE\_COMMAND = 0x32,**

* ???? // KISS frames

**CRSF\_FRAMETYPE\_KISS\_REQ = 0x78,**

* ????

**CRSF\_FRAMETYPE\_KISS\_RESP = 0x79,**

* ???? // MSP commands

**CRSF\_FRAMETYPE\_MSP\_REQ = 0x7A,**

* ????

**CRSF\_FRAMETYPE\_MSP\_RESP = 0x7B,**

* ????

**CRSF\_FRAMETYPE\_MSP\_WRITE = 0x7C,**

* ????