Uart Protocol Between FC And Receiver

The SR12S or satellite receiver communicates with the FC through a UART and its baud speed is 115200, 1 stop bit, no parity. Satellite sends packet to FC or FC to Satellite in 50Hz.

**Data structure we use:**

typedef struct{

uint8\_t header1;

uint8\_t header2;

uint8\_t length;

uint8\_t type;

uint8\_t data[];

uint8\_t crc8;

} ReceiverFcPacket;

1)The values of header1 and the header2 are 0x55 .

2)The length includes type, data[], crc8 and does not include itself. So length = sizeof(type)+sizeof(data[])+sizeof(crc8).

3)The definition of type is 0X00=ChannelData12, 0X01=ChannelData24, 0X02=TelemetryData, 0X03=ChannelData12\_Gps, 0X04=Bind\_StBindCmd, ..., and so on. These packets are talked latter. We can increase the number by 1 each time we add new packet type.

4) The data[] is said below.

5) The crc8 is calculated from the length to the last byte of data[].

**A. The definition of the data[] for the packet sent from satellite to the FC.**

typedef struct{

uint16\_t t; //packet counter or clock

uint8\_t status[2]; //status flags, signal strength

uint8\_t channel[18]; //12 channel data (compressed)

} ChannelData12;

typedef struct{

uint16\_t t; //packet counter or clock

uint8\_t status[2]; //status flags, signal strength

uint8\_t channel[18]; //12 channel data (compressed)

uint32\_t lat;// need to be divided by 10^7 to get the actual latitude

uint32\_t lon; // need to be divided by 10^7 to get the actual lon

float alt;

int16\_t accuracy;

int16\_t speed; //over ground speed

int16\_t angle;

uint8\_t fixmode\_satellite; //

float compass;

} ChannelData12\_Gps;

1) The resolution of the t is 0.1ms.

2) Status[0] is the RX signal strength and the status[1] is the counts of the UART packets that receiver has sent to the FC since the satellite received the latest RF packet.

3)The following is the way to compress the channel data. For example , if channeldata[0]=0x123, channeldata[1]=0x456, channeldata[2]= 0x789,……, and the channel[] is channel[0]=0x12, channel[1]=0x34; channel [2]=0x56, channel [3]=0x78, ……

4)The lowest 5 bits of the fixmode\_satellite is for number of the satellite(GPS in the ST24), and the highest 3 bits are reserved for fix mode.

5)The ChannelData12 and the ChannelData12\_Gps sent from satellite to the FC in 50Hz, ChannelData12 is in 45Hz and ChannelData12\_Gps in 5Hz.

**B. The definition of the data[] for the packet sent from the FC to satellite.**

typedef struct{

uint16\_t t;

uint8\_t cmd[4];

}StBindCmd;

1. StBindCmd is used by the FC to tell satellite to bind, if FC wants to be bint. After receiving the bind command, the satellite will go to bind mode, and it will quit bind mode automatically if it is bint successfully.

2)cmd[0]=”B”, cmd[1]=”I”,cmd[2]=”N”,cmd[3]=”D”.

typedef struct{

uint16\_t t; //packet counter or clock

int32\_t lat; //lattitude (degrees) +/- 90 deg

int32\_t lon; //longitude (degrees) +/- 180 deg

int32\_t alt; // 0.01m resolution, altitude (meters)

int16\_t vx,vy,vz; //velocity 0.01m res, +/-320.00 North-East- Down

uint8\_t nsat; //number of satellites

uint8\_t voltage; //25.4V voltage = 5 + 255\*0.1 = 30.5V, min=5V

uint8\_t current; //0.5A resolution

int16\_t roll, pitch, yaw; //0.01 degree resolution

uint8\_t motorStatus; //1 bit per motor for status 1=good, 0= fail

uint8\_t imuStatus;

uint8\_t pressCompassStatus;

} TelemetryData; //32 bytes

1. The resolution of the t is 0.1ms.
2. The definition of imuStatus:

8 bit total

bits 0-2 for status

- value 0 is FAILED

- value 1 is INITIALIZING

- value 2 is RUNNING

- values 3 through 7 are reserved

bits 3-7 are status for sensors (0 or 1)

- mpu6050

- accelerometer

- primary gyro x

- primary gyro y

- primary gyro z

1. The definition of pressCompassStatus :

pressCompassStatus

8 bit total

bits 0-3 for compass status

- value 0 is FAILED

- value 1 is INITIALIZING

- value 2 is RUNNING

- value 3 - 15 are reserved

bits 4-7 for pressure status

- value 0 is FAILED

- value 1 is INITIALIZING

- value 2 is RUNNING

- value 3 - 15 are reserved

**C. The crc8 algorithm:**

uint8 common\_crc8(uint8 \*ptr, uint8 len)

{

uint8 i,crc ;

crc = 0;

while(len--)

{

for(i = 0x80; i != 0; i >>= 1)

{

if((crc & 0x80) != 0)

{

crc <<= 1;

crc ^= 0x07;

}

else

{

crc <<= 1;

}

if((\*ptr & i) != 0)

{

crc ^= 0x07;

}

}

ptr++;

}

return (crc);

}

crc\_table:

**#define** ST24\_CRC {\

0x00, 0x07, 0x0E, 0x09, 0x1C, 0x1B, 0x12, 0x15, 0x38, 0x3F, 0x36, 0x31, \

0x24, 0x23, 0x2A, 0x2D, 0x70, 0x77, 0x7E, 0x79, 0x6C, 0x6B, 0x62, 0x65, \

0x48, 0x4F, 0x46, 0x41, 0x54, 0x53, 0x5A, 0x5D, 0xE0, 0xE7, 0xEE, 0xE9, \

0xFC, 0xFB, 0xF2, 0xF5, 0xD8, 0xDF, 0xD6, 0xD1, 0xC4, 0xC3, 0xCA, 0xCD, \

0x90, 0x97, 0x9E, 0x99, 0x8C, 0x8B, 0x82, 0x85, 0xA8, 0xAF, 0xA6, 0xA1,\

0xB4, 0xB3, 0xBA, 0xBD, 0xC7, 0xC0, 0xC9, 0xCE, 0xDB, 0xDC, 0xD5, 0xD2,\

0xFF, 0xF8, 0xF1, 0xF6, 0xE3, 0xE4, 0xED, 0xEA, 0xB7, 0xB0, 0xB9, 0xBE, 0xAB, \

0xAC, 0xA5, 0xA2, 0x8F, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9D, 0x9A, 0x27, 0x20, \

0x29, 0x2E, 0x3B, 0x3C, 0x35, 0x32, 0x1F, 0x18, 0x11, 0x16, 0x03, 0x04, 0x0D, \

0x0A, 0x57, 0x50, 0x59, 0x5E, 0x4B, 0x4C, 0x45, 0x42, 0x6F, 0x68, 0x61, 0x66, \

0x73, 0x74, 0x7D, 0x7A, 0x89, 0x8E, 0x87, 0x80, 0x95, 0x92, 0x9B, 0x9C, 0xB1,\

0xB6, 0xBF, 0xB8, 0xAD, 0xAA, 0xA3, 0xA4, 0xF9, 0xFE, 0xF7, 0xF0, 0xE5, 0xE2, \

0xEB, 0xEC, 0xC1, 0xC6, 0xCF, 0xC8, 0xDD, 0xDA, 0xD3, 0xD4, 0x69, 0x6E, 0x67,\

0x60, 0x75, 0x72, 0x7B, 0x7C, 0x51, 0x56, 0x5F, 0x58, 0x4D, 0x4A, 0x43, 0x44, 0x19,\

0x1E, 0x17, 0x10, 0x05, 0x02, 0x0B, 0x0C, 0x21, 0x26, 0x2F, 0x28, 0x3D, 0x3A, 0x33,\

0x34, 0x4E, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5C, 0x5B, 0x76, 0x71, 0x78, 0x7F, 0x6A, \

0x6D, 0x64, 0x63, 0x3E, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2C, 0x2B, 0x06, 0x01, 0x08, \

0x0F, 0x1A, 0x1D, 0x14, 0x13, 0xAE, 0xA9, 0xA0, 0xA7, 0xB2, 0xB5, 0xBC, 0xBB, 0x96, \

0x91, 0x98, 0x9F, 0x8A, 0x8D, 0x84, 0x83, 0xDE, 0xD9, 0xD0, 0xD7, 0xC2, 0xC5, 0xCC, \

0xCB, 0xE6, 0xE1, 0xE8, 0xEF, 0xFA, 0xFD, 0xF4, 0xF3\

}

**const** uint8\_t crcTable[256] = ST24\_CRC;

uint8\_t **st24\_common\_crc8**(uint8\_t \*ptr, uint8\_t len)

{

uint8\_t crc = 0;

**for** (uint8\_t i = 0; i < len; i++) {

crc = crcTable[crc^\*ptr++];

}

**return** crc;

}