通讯数据格式

|  |  |  |  |
| --- | --- | --- | --- |
| Frame\_header(4字节) | Cmd\_id（2字节） | Data(n字节) | Frame\_tail(2字节CRC16，整包校验) |

Frame\_header格式

|  |  |  |
| --- | --- | --- |
| SOF | Data\_length | CRC8 |
| 1字节 | 2字节 | 1字节 |

Frame\_header详细定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 偏移 | 大小（字节） | 功能描述 |
| SOF | 0 | 1 | 起始字节，固定值位0xA5 |
| Data\_length | 1 | 2 | 数据帧中data长度 |
| CRC8 | 3 | 1 | 帧头CRC8校验 |
|  |  |  |  |

原来的通讯格式中有一个seq，包序号，我觉得没什么用就删掉了

cmd\_id命令码id说明

|  |  |  |
| --- | --- | --- |
| 命令码 | 数据段长度（字节） | 功能说明 |
| 0x0001 | 13 | 转发遥控器控制数据 |

未完待续

0x0001

**转发遥控器控制数据**

**#define CMD\_SBUS (0x0001)**

**//#define MODE\_STOP 0 //停止模式，**

**//#define MODE\_NORMAL 1 //正常模式**

**//#define MODE\_BACK\_ZERO 2;//回零模式**

**enum{**

**MODE\_STOP = 0,**

**MODE\_NORMAL,**

**MODE\_BACK\_ZERO**

**};**

**struct Command\_Sbus{**

**uint8\_t mode;**

**//目前只用了低四位，高四位还可以利用一下**

**//0:MODE\_STOP停止模式，**

**//1:MODE\_NORMAL正常模式**

**//2:MODE\_BACK\_ZERO**

**int32\_t vx;**

**int32\_t vy;**

**int32\_t vw;**

**};//13个字节**

相关代码例子：

**发送函数**

void send2PC(uint16\_t CMD,uint8\_t \*data,uint8\_t len)

{

// uint8\_t\* tmp;

Header.data\_length=len;

uint8\_t buf[128];

uint8\_t index=0;

uint8\_t len\_tmp;

Append\_CRC8\_Check\_Sum( (uint8\_t \*)(&Header),4);

len\_tmp=sizeof(Header);

memcpy(buf,&Header,len\_tmp);

index+=len\_tmp; //偏移到cmd起始位置

len\_tmp=sizeof(CMD);

memcpy(buf+index,&CMD,len\_tmp);

index+=len\_tmp; //偏移到data起始位置

memcpy(buf+index,data,len);

index+=len;//偏移到CRC16的起始位置

Append\_CRC16\_Check\_Sum(buf,index+2);//这里的长度是整个数据的长度，而不只是data的长度

index+=2;//这里的index也是长度

tmp\_fuck=index;

HAL\_GPIO\_WritePin(GPIOH,GPIO\_PIN\_8,GPIO\_PIN\_SET);//开启485使能

HAL\_UART\_Transmit(&huart3,buf,index,5);

HAL\_GPIO\_WritePin(GPIOH,GPIO\_PIN\_8,GPIO\_PIN\_RESET);//关闭485使能

}

**CRC校验**

CRC8校验，直接照抄

//crc8 generator polynomial:G(x)=x8+x5+x4+1

const unsigned char CRC8\_INIT = 0xff;

const unsigned char CRC8\_TAB[256] =

{

0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83, 0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,

0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e, 0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc, 0x23,

0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0, 0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62, 0xbe, 0xe0,

0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d, 0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff, 0x46, 0x18, 0xfa,

0xa4, 0x27, 0x79, 0x9b, 0xc5, 0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07, 0xdb, 0x85, 0x67,

0x39, 0xba, 0xe4, 0x06, 0x58, 0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a, 0x65, 0x3b, 0xd9, 0x87,

0x04, 0x5a, 0xb8, 0xe6, 0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24, 0xf8, 0xa6, 0x44, 0x1a, 0x99,

0xc7, 0x25, 0x7b, 0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,

0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f, 0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd, 0x11,

0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92, 0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50, 0xaf, 0xf1,

0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c, 0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee, 0x32, 0x6c, 0x8e,

0xd0, 0x53, 0x0d, 0xef, 0xb1, 0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73, 0xca, 0x94, 0x76, 0x28,

0xab, 0xf5, 0x17, 0x49, 0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b, 0x57, 0x09, 0xeb, 0xb5,

0x36, 0x68, 0x8a, 0xd4, 0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16, 0xe9, 0xb7, 0x55, 0x0b, 0x88,

0xd6, 0x34, 0x6a, 0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,

0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7, 0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35,

};

unsigned char Get\_CRC8\_Check\_Sum(unsigned char \*pchMessage,unsigned int dwLength,unsigned char ucCRC8)

{

unsigned char ucIndex;

while (dwLength--)

{

ucIndex = ucCRC8^(\*pchMessage++);

ucCRC8 = CRC8\_TAB[ucIndex];

}

return(ucCRC8);

}

/\*

\*\* Descriptions: CRC8 Verify function

\*\* Input: Data to Verify,Stream length = Data + checksum

\*\* Output: True or False (CRC Verify Result)

这个函数是验证接收数据的CRC

\*/

unsigned int Verify\_CRC8\_Check\_Sum(unsigned char \*pchMessage, unsigned int dwLength) //这里输入的长度是包括CRC校验码的长度

{

unsigned char ucExpected = 0;

if ((pchMessage == 0) || (dwLength <= 2)) return 0;

ucExpected = Get\_CRC8\_Check\_Sum (pchMessage, dwLength-1, CRC8\_INIT);

return ( ucExpected == pchMessage[dwLength-1] );

}

/\*

\*\* Descriptions: append CRC8 to the end of data

\*\* Input: Data to CRC and append,Stream length = Data + checksum

\*\* Output: True or False (CRC Verify Result)

将数据设置CRC校验码插入到数据的尾部

\*/

void Append\_CRC8\_Check\_Sum(unsigned char \*pchMessage, unsigned int dwLength) //这里输入的长度是包括CRC校验码的长度

{

unsigned char ucCRC = 0;

if ((pchMessage == 0) || (dwLength <= 2))

return;

ucCRC = Get\_CRC8\_Check\_Sum ( (unsigned char \*)pchMessage, dwLength-1, CRC8\_INIT);

pchMessage[dwLength-1] = ucCRC;

}

同理，CRC16校验如下：

uint16\_t CRC\_INIT = 0xffff;

const uint16\_t wCRC\_Table[256] =

{

0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,

0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,

0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,

0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,

0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,

0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,

0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,

0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,

0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,

0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,

0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,

0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,

0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,

0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,

0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,

0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,

0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,

0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,

0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,

0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,

0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,

0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,

0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,

0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,

0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,

0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,

0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,

0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,

0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,

0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,

0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,

0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78

};

/\*

\*\* Descriptions: CRC16 checksum function

\*\* Input: Data to check,Stream length, initialized checksum

\*\* Output: CRC checksum

\*/

uint16\_t Get\_CRC16\_Check\_Sum(uint8\_t \*pchMessage,uint32\_t dwLength,uint16\_t wCRC)

{

uint8\_t chData;

if (pchMessage == NULL)

{

return 0xFFFF;

}

while(dwLength--)

{

chData = \*pchMessage++;

(wCRC) = ((uint16\_t)(wCRC) >> 8) ^ wCRC\_Table[((uint16\_t)(wCRC) ^ (uint16\_t)(chData)) & 0x00ff];

}

return wCRC;

}

/\*

\*\* Descriptions: CRC16 Verify function

\*\* Input: Data to Verify,Stream length = Data + checksum

\*\* Output: True or False (CRC Verify Result)

\*/

uint32\_t Verify\_CRC16\_Check\_Sum(uint8\_t \*pchMessage, uint32\_t dwLength)

{

uint16\_t wExpected = 0;

if ((pchMessage == NULL) || (dwLength <= 2))

{

//return \_\_FALSE;

return 0;

}

wExpected = Get\_CRC16\_Check\_Sum ( pchMessage, dwLength - 2, CRC\_INIT);

return ((wExpected & 0xff) == pchMessage[dwLength - 2] && ((wExpected >> 8) & 0xff) ==

pchMessage[dwLength - 1]);

}

/\*

\*\* Descriptions: append CRC16 to the end of data

\*\* Input: Data to CRC and append,Stream length = Data + checksum

\*\* Output: True or False (CRC Verify Result)

\*/

void Append\_CRC16\_Check\_Sum(uint8\_t \* pchMessage,uint32\_t dwLength)

{

uint16\_t wCRC = 0;

if ((pchMessage == NULL) || (dwLength <= 2))

{

return;

}

wCRC = Get\_CRC16\_Check\_Sum ( (uint8\_t \*)pchMessage, dwLength-2, CRC\_INIT );

pchMessage[dwLength-2] = (uint8\_t)(wCRC & 0x00ff);

pchMessage[dwLength-1] = (uint8\_t)((wCRC >> 8)& 0x00ff);

}