nRF52832 与 RV1126串口通信协议

串口通信由一条一条的命令实现，命令结构体(见nrf\_rv1126\_uart\_t结构体定义)由三部分组成：4字节起始码，即0xFFFFFFFF；2字节具体命令码(见nrf\_rv1126\_cmd\_t枚举类型定义)；2字节CRC校验码。

1．宏定义

#define NRU\_START\_CODE 0xFFFFFFFF

#define NRC\_LEN sizeof(nrf\_rv1126\_uart\_t)

#define CRC\_LEN sizeof(uint16\_t)

2. 结构体

/\* nRF52 与 rv1126 UART 通信结构体定义 \*/

typedef struct

{

uint32\_t start\_code; // 即NRU\_START\_CODE，一条命令的起始标志

uint16\_t cmd;

uint16\_t crc; // 16位CRC校验

}nrf\_rv1126\_uart\_t;

3. 枚举类型

/\* nRF52与rv1126 UART通信命令枚举类型定义，即上述结构体的第二个字段 \*/

typedef enum

{

/\* nRF52832 to rv1126 command type \*/

N2R\_CMD\_TYPE\_NEXT\_PARA = 0, // 下一段

N2R\_CMD\_TYPE\_PHOTO, // 拍照

N2R\_CMD\_TYPE\_PREV\_PARA, // 上一段

N2R\_CMD\_TYPE\_PAUSE, // 暂停

N2R\_CMD\_TYPE\_MINUS, // "-"

N2R\_CMD\_TYPE\_PLUS, // "+"

N2R\_CMD\_TYPE\_MODE, // 模式

N2R\_CMD\_TYPE\_BOND\_SUCCESS, // BLE配对成功

N2R\_CMD\_TYPE\_DEL\_BONDS\_SUCCESS, // 擦除BLE配对信息成功

/\* rv1126 to nRF52832 command type \*/

R2N\_CMD\_TYPE\_DELETE\_BONDS = 0x80, //擦除nRF52832的BLE bonds信息

}nrf\_rv1126\_cmd\_t;

4. CRC检验实现

static uint16\_t crc16\_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

};

uint16\_t get\_crc\_value(uint8\_t \* data, uint32\_t length)

{

uint16\_t crc\_reg = 0xFFFF;

while (length--)

{

crc\_reg = (crc\_reg >> 8) ^ crc16\_table[(crc\_reg ^ \*data++) & 0xff];

}

return (uint16\_t)(~crc\_reg) & 0x0000FFFF;

}

uint16\_t crc\_value;

bool crc\_check\_do(uint8\_t \* data, uint32\_t length, uint16\_t right\_crc\_value)

{

crc\_value = get\_crc\_value(data, length);

if (crc\_value != right\_crc\_value)

return false;

else

return true;

}