```
#include "bootstart.h"
#include "upfile.h"
#include "boardparents.h"
#include "nrf_nvmc.h"
const uint8_t hardversion[8] __attribute__ ((at(0x18100))) = {PCB_VERSION,0x10,0x34,0x56,0xcc,0,0,0}; /**< This variable ensures that the linker script will write the bootloader start address to the UICR register. This value will be written in the HEX file and thus written to UICR when the bootloader is flashed into the chip. */
void V_FeedWdog(void);
void DisableGPIOEInt(void);
uint8_t rfidsendbuf[32],rfidRev[32];
bool RevOk=false;
extern uint8_t m_addl_adv_manuf_data[6];
  void RevRfDataRecall(uint8_t * datain);
static void AskPacket(uint16_t packet);
static void AskCheckSum(void);
uint32_t fileOffset,filesize;
uint8_t IapFlg[12];
/*
2.4g 更新程序
*/
extern u16 wire_work_sec;
void TaskUpFile(void)
{

      ChangeToEsb(true,RevRfDataRecall);/*进入 rf 模式*/
      IntoTestMode();
      ChangeToEsb(false,NULL);/*返回蓝牙模式*/

    if(wire_work_sec==1)
    {
      ChangeToEsb(true,RevRfDataRecall);/*进入 rf 模式*/
      IntoTestMode();
      RfOn(false);
   }
}
/*
*****************************************************************************
*************************
*    函 数 名:void Talk10cmRevDataRecall(u8* datain)
*    功能说明:rf 收到数据后回调
*
*          作      者 ： liupeng
```

```
*    形      参：
*          版      本：version 1.0
*    返 回 值:
*******************************************************************************
*************************
*/
 void RevRfDataRecall(uint8_t * datain)
{
      if(NRF_RADIO->CRCSTATUS != 1) return;

      if(0==memcmp(&rfidsendbuf[1],&datain[1],4))
      {
            memcpy(rfidRev,datain,30);
            RevOk=true;

      }
}
bool SendPack(uint16_t timeout,uint8_t len)
{
      uint16_t tmp=0;
      RevOk=false;
      RfTx(rfidsendbuf,len);/*adv*/
      while( (tmp++<timeout)&&(true!=RevOk));
      return RevOk;

}
/*
内部 flash 操作
*/

void EraseBvkFlash(uint32_t size)
{
uint32_t num=0,i,addr;
num=size/1024;
if(size%1024)
{
num++;
}
addr=APP_BVK_ADDR/1024;
for(i=0;i<num;i++)
{
Flash_Erase_Page(addr+i);
}
Flash_Erase_Page(BOOT_PARA_ADDR/1024);
```

```
}
void WriteBvkFlash(uint32_t* buf,uint32_t numOfDword)
{
	Flash_Write_World((uint32_t *)(APP_BVK_ADDR+fileOffset),buf,numOfDword);
	fileOffset+=(numOfDword*4);
}
static void interrupts_disable(void)
{
	uint32_t interrupt_setting_mask;
	uint8_t	irq;

	// We start the loop from first interrupt, i.e. interrupt 0.
	irq 				= 0;
	// Fetch the current interrupt settings.
	interrupt_setting_mask = NVIC->ISER[0];

#define IRQ_ENABLED			0x01			/**< Field identifying if an
interrupt is enabled. */
#define MAX_NUMBER_INTERRUPTS	32			/**< Maximum number of
interrupts available. */

	for (; irq < MAX_NUMBER_INTERRUPTS; irq++)
	{
		if (interrupt_setting_mask & (IRQ_ENABLED << irq))
		{
			// The interrupt was enabled, and hence disable it.
			NVIC_DisableIRQ((IRQn_Type) irq);
		}
	}
}
void WriteFlgBootSys(void)
{
	uint32_t buf[3];
	memcpy(buf,IapFlg,12);
	Flash_Write_World(((uint32_t *)BOOT_PARA_ADDR),(uint32_t *)buf,3);
	DisableGPIOEInt();
	interrupts_disable();
   // NVIC_SystemReset();
	V_FeedWdog();
	StartApplication(BOOT_ADDR);
}
bool RfWritePacket(uint8_t*rev_buf,uint16_t maxbuf,uint16_t *rev_lenth,uint32_t *packet )
{
```

```
uint8_t write_all;

        write_all=false;
        memcpy(&rev_buf[rev_lenth[0]],&rfidRev[3],16);
        rev_lenth[0]+=16;
        if((fileOffset+rev_lenth[0])<filesize)
        {
        packet[0]++;
        AskPacket(packet[0]);
        }

        if(rev_lenth[0]>=maxbuf)
        {
        //IapWrite(rev_buf,rev_lenth[0]);
        WriteBvkFlash((uint32_t*)rev_buf,rev_lenth[0]/4);
        rev_lenth[0]=0;
        }

        if((fileOffset+rev_lenth[0])>=filesize)
        {
                if(rev_lenth[0])
                {
                //IapWrite(rev_buf,rev_lenth[0]);
                WriteBvkFlash((uint32_t*)rev_buf,rev_lenth[0]/4);
                rev_lenth[0]=0;
                }
                AskCheckSum();
                write_all=true;

        }

      return write_all;
}
void RfRevFileRecall(uint8_t *revbuf)
{
        if(NRF_RADIO->CRCSTATUS != 1) return;


        if(RevOk==true) return;
          if((revbuf[0]==UPFILE_DATA_24)||\
                (revbuf[0]==UPFILE_CHECKSUM_24)||\
                (revbuf[0]==UPFILE_END_24))
                {
                        if(revbuf[19]==CheckSum(revbuf,19))
```

```
                {
                memcpy(rfidRev,revbuf,30);
                RevOk=true;
                }
            }
}

static void AskPacket(uint16_t packet)
{
    RevOk=false;
    if(0==packet)
    {
//          rfidsendbuf[0]=0x20;
//          memcpy(&rfidsendbuf[1],StuId.deviceid,4);
//          rfidsendbuf[5]=CheckSum(rfidsendbuf,5);
            rfidsendbuf[6]=UPFILE_DATA_24;
    }
    rfidsendbuf[7]=packet>>8;
    rfidsendbuf[8]=packet;
    rfidsendbuf[9]=CheckSum(rfidsendbuf,9);
    RfTx(rfidsendbuf,20);/*adv*/
}

static void AskCheckSum(void)
{
    RevOk=false;

    rfidsendbuf[0]=0x20;
//  memcpy(&rfidsendbuf[1],StuId.deviceid,4);
//  rfidsendbuf[5]=CheckSum(rfidsendbuf,5);
    rfidsendbuf[6]=UPFILE_CHECKSUM_24;
    rfidsendbuf[7]=0;
    rfidsendbuf[8]=1;
    //rfidsendbuf[8]=0;
    rfidsendbuf[9]=CheckSum(rfidsendbuf,9);
    RfTx(rfidsendbuf,20);/*adv*/

}

bool IsRightBoard(char* boardbuf)
{
    bool OkFlg=true;
    if(0!=memcmp((char*)&hardversion,boardbuf,5))
        OkFlg=false;
```

```
	return OkFlg;

}
uint16_t CalcCrc16(uint16_t crc,const uint8_t * dat,uint32_t len)
{
	uint8_t    da;
	uint8_t    Data_Temp = 0;

	while(len--!=0)
	{
		da=(unsigned char) (crc/256);
		crc <<= 8;
		Data_Temp = *dat;
		crc ^= crc16_table[da^Data_Temp];
		dat++;
	}
	return crc;
}

bool GetFileCrc(uint16_t crcin,uint16_t* crcout)
{

uint16_t tmpcrc=0xffff;
crcout[0]=CalcCrc16(tmpcrc,(const uint8_t *)APP_BVK_ADDR,filesize);
if(crcin==crcout[0])
return true;
return false;

}

void RfRevFileSendResult(uint16_t realcrc)
{
	RevOk=false;

	rfidsendbuf[0]=0x20;
//	memcpy(&rfidsendbuf[1],StuId.deviceid,4);
//	rfidsendbuf[5]=CheckSum(rfidsendbuf,5);
	rfidsendbuf[6]=UPFILE_END_24;

	rfidsendbuf[7]=realcrc>>8;
	rfidsendbuf[8]=realcrc;

	rfidsendbuf[9]=CheckSum(rfidsendbuf,9);
```

```
	RfTx(rfidsendbuf,20);/*adv*/

}
void TaskP24UpdataFile(uint8_t fre)
{
#define MAX_REV_BUF_LEN 256
uint32_t packet=0,revpacket;
uint32_t ticktime=0;
uint16_t crc,crcout;
uint8_t tick20ms;
bool crc_ok=false;
uint16_t timeout10s=0;
uint8_t rev_buf[MAX_REV_BUF_LEN+16];
uint16_t rev_lenth;


#define sec3_IN5ms	600
#define TIMEOUT5ms	(TICK_TO_MS/200)

#define STA_ASK_DATA	0x00
#define STA_ASK_CHECKSUM	0x01
#define STA_SEND_RESULT	0x02

uint8_t sta=STA_ASK_DATA;
if(fre<5)
Nrf51Config_FUN(30,20,16,RfRevFileRecall);
else
Nrf51Config_FUN(fre,20,16,RfRevFileRecall);
ticktime=NRF_RTC1->COUNTER;
tick20ms=0;
rev_lenth=0;
AskPacket(packet);

	while(1)
	{
		if(RevOk==true)
		{
		   if(rfidRev[0]==UPFILE_DATA_24)
			{
				timeout10s=0;
				revpacket=rfidRev[1];
				revpacket=(revpacket<<8)+rfidRev[2];
				if(revpacket==packet)
				{
```

```
                    if((packet==16)&&\
                        (true!=IsRightBoard((char*)&rfidRev[3])))
                    {
                        /*文件错误*/
                        RfRevFileSendResult(0);
                        return;
                    }
                    tick20ms=0;

if(true==(RfWritePacket(rev_buf,MAX_REV_BUF_LEN,&rev_lenth,&packet)))
                    {
                    sta=STA_ASK_CHECKSUM;
                    }

                }else
                {
                RevOk=false;
                }
            }else   if((rfidRev[0]==UPFILE_CHECKSUM_24)&&
            (0==memcmp(&rfidRev[6],&rfidsendbuf[1],4)))
            {
                    crc=rfidRev[1];
                    crc=(crc<<8)+rfidRev[2];
                    crc_ok=GetFileCrc(crc,&crcout);
                    IapFlg[0]=0xaa;
                    IapFlg[1]=0x55;
                    IapFlg[2]=filesize>>24;
                    IapFlg[3]=filesize>>16;
                    IapFlg[4]=filesize>>8;
                    IapFlg[5]=filesize;
                    IapFlg[6]=0;//version
                    IapFlg[7]=0;//version
                    IapFlg[8]=crcout>>8;
                    IapFlg[9]=crcout>>0;
                    RfRevFileSendResult(crcout);
                    sta=STA_SEND_RESULT;
                    timeout10s=0;
                    tick20ms=0;


            }else   if(rfidRev[0]==UPFILE_END_24)
            {
                if(true==crc_ok)
                {
```

```
                    WriteFlgBootSys();
                }
                timeout10s=0;
                tick20ms=0;
                return ;
            }

    }


    V_FeedWdog();
  if(TickEscape(NRF_RTC1->COUNTER,ticktime,0x00ffffff)<TIMEOUT5ms)
    {
    continue;
    }
  ticktime=NRF_RTC1->COUNTER;
tick20ms++;
  if(tick20ms>3)
    {
        tick20ms=0;
        if(sta==STA_ASK_DATA)
         {
              AskPacket(packet);
         }
        else    if(sta==STA_ASK_CHECKSUM)
        {
             AskCheckSum();
        } else    if(sta==STA_SEND_RESULT)
        {
         RfRevFileSendResult(crcout);
        }
    }


    if(timeout10s++>sec3_IN5ms)
    {
         if(true==crc_ok)
         {
              WriteFlgBootSys();
         }

         return;
    }
```

```
        }



}


void IntoTestMode(void)
{
        uint8_t *ptdata;
        uint32_t size;
        rfidsendbuf[0]=0x20;
        rfidsendbuf[1]=0x0b;
        rfidsendbuf[2]=0x00;
        rfidsendbuf[3]=m_addl_adv_manuf_data[4];
        rfidsendbuf[4]=m_addl_adv_manuf_data[5];
        rfidsendbuf[5]=CheckSum(rfidsendbuf,5);

          rfidsendbuf[6]=SOFT_VERSION/10;
        rfidsendbuf[7]=SOFT_VERSION%10;
        rfidsendbuf[8]=((PCB_VERSION/10)<<4)|(PCB_VERSION%10);

          rfidsendbuf[9]=0;

        if(false == SendPack(400,10)) return;
        if(rfidRev[9]!=CheckSum(rfidRev,9)) return;
        ptdata=&rfidRev[5];
        switch(rfidRev[0])
        {
                case UPFILE_START_24:
                size=ptdata[1];
                size=(size<<8)+ptdata[2];
                size=(size<<8)+ptdata[3];
                filesize=size;
                fileOffset=0;
                EraseBvkFlash(size);

                TaskP24UpdataFile(ptdata[0]);
                break;
                case READ_MAC_24 :
                Delay_100us(10);
                rfidsendbuf[0]=rfidRev[0];
                memcpy(&rfidsendbuf[1],m_addl_adv_manuf_data,6);
```

```
            rfidsendbuf[7]=rfidRev[7];
            rfidsendbuf[8]=rfidRev[8];
            rfidsendbuf[9]=CheckSum(&rfidsendbuf[0],9);
            RfTx(rfidsendbuf,10);
            break;
            default:
            break;
        }


}
```