**安全帽V1.0**

**源代码**

#include "sys.h"

#include "STM32F4\_Head.h"

#include "PIONEER\_Config.h"

#include "const.h"

#include "APP\_Init.h"

void TIM\_Init(void)

{

#if (TIM2\_EN > 0)

TIM2\_Init();

TIM\_Cmd(TIM2, DISABLE);

UART1\_RX\_STA=0;

#endif

#if (TIM3\_EN > 0)

TIM3\_Init();

TIM\_Cmd(TIM3, DISABLE);

UART2\_RX\_STA=0;

#endif

#if (TIM4\_EN > 0)

TIM4\_Init();

TIM\_Cmd(TIM4, DISABLE);

UART3\_RX\_STA=0;

#endif

#if (TIM5\_EN > 0)

TIM5\_Init();

TIM\_Cmd(TIM5, DISABLE);

UART4\_RX\_STA=0;

#endif

#if (TIM6\_EN > 0)

TIM6\_Init();

TIM\_Cmd(TIM6, DISABLE);

UART5\_RX\_STA=0;

#endif

#if (TIM7\_EN > 0)

TIM7\_Init();

TIM\_Cmd(TIM7, DISABLE);

UART6\_RX\_STA=0;

#endif

}

void Uart\_Init(void)

{

#if (UART1\_EN > 0)

Uart1\_Init();

#endif

#if (UART2\_EN > 0)

Uart2\_Init();

#endif

#if (UART3\_EN > 0)

Uart3\_Init();

#endif

#if (UART4\_EN > 0)

Uart4\_Init();

#endif

#if (UART5\_EN > 0)

Uart5\_Init();

#endif

#if (UART6\_EN > 0)

Uart6\_Init();

#endif

#if ((UART1\_RX\_MODE == 2)||(UART2\_RX\_MODE == 2)||(UART3\_RX\_MODE == 2)||(UART4\_RX\_MODE == 2)||(UART5\_RX\_MODE == 2)||(UART6\_RX\_MODE == 2))

#endif

}

void IO\_Init(void)

{

IN\_Init();

OUT\_Init();

}

#if (RTC\_EN > 0)

#define RTC\_PWROFF\_FLAG 0x5050

void RTC\_APPInit(void)

{

RTC\_TimeTypeDef RTC\_Time;

RTC\_DateTypeDef RTC\_Date;

INT8U flag;

flag = RTCInit();

if (flag == ERR\_TRUE)

{

#if (DEBUG\_APP\_EN == 1)

printf("RTC³õÊ¼»¯: OK\r\n");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("RTC³õÊ¼»¯: ERROR\r\n");

#endif

}

if(RTC\_ReadBackupRegister(RTC\_BKP\_DR0)!=RTC\_PWROFF\_FLAG)

{

RTC\_Date.RTC\_Year = 20;

RTC\_Date.RTC\_Month = 1;

RTC\_Date.RTC\_Date = 15;

RTC\_Date.RTC\_WeekDay = 3;

RTC\_Time.RTC\_H12 = RTC\_H12\_AM;

RTC\_Time.RTC\_Hours = 15;

RTC\_Time.RTC\_Minutes = 28;

RTC\_Time.RTC\_Seconds = 56;

RTC\_SetDate(RTC\_Format\_BIN,&RTC\_Date);

RTC\_SetTime(RTC\_Format\_BIN,&RTC\_Time);

RTC\_WriteBackupRegister(RTC\_BKP\_DR0,RTC\_PWROFF\_FLAG);

}

#if (RTC\_ALRAIT\_EN > 0)

RTC\_Set\_Alarm(RTC\_Alarm\_A,2,5,30,0);

#endif

#if (RTC\_ALRBIT\_EN > 0)

RTC\_Set\_Alarm(RTC\_Alarm\_B,2,5,30,0);

#endif

#if (RTC\_WKUPIT\_EN > 0)

RTC\_Set\_WakeUp(RTC\_WakeUpClock\_CK\_SPRE\_16bits,1);

#endif

}

#endif

void EXTI\_APPInit(void)

{

EXTI\_PARA Para;

#if (EXTI0\_EN > 0)

Para.id = EXTI0\_ID;

Para.Mode = EXTI0\_MODE;

Para.Port = EXTI0\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI1\_EN > 0)

Para.id = EXTI1\_ID;

Para.Mode = EXTI1\_MODE;

Para.Port = EXTI1\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI2\_EN > 0)

Para.id = EXTI2\_ID;

Para.Mode = EXTI2\_MODE;

Para.Port = EXTI2\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI3\_EN > 0)

Para.id = EXTI3\_ID;

Para.Mode = EXTI3\_MODE;

Para.Port = EXTI3\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI4\_EN > 0)

Para.id = EXTI4\_ID;

Para.Mode = EXTI4\_MODE;

Para.Port = EXTI4\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI5\_EN > 0)

Para.id = EXTI5\_ID;

Para.Mode = EXTI5\_MODE;

Para.Port = EXTI5\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI6\_EN > 0)

Para.id = EXTI6\_ID;

Para.Mode = EXTI6\_MODE;

Para.Port = EXTI6\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI7\_EN > 0)

Para.id = EXTI7\_ID;

Para.Mode = EXTI7\_MODE;

Para.Port = EXTI7\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI8\_EN > 0)

Para.id = EXTI8\_ID;

Para.Mode = EXTI8\_MODE;

Para.Port = EXTI8\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI9\_EN > 0)

Para.id = EXTI9\_ID;

Para.Mode = EXTI9\_MODE;

Para.Port = EXTI9\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI10\_EN > 0)

Para.id = EXTI10\_ID;

Para.Mode = EXTI10\_MODE;

Para.Port = EXTI10\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI11\_EN > 0)

Para.id = EXTI11\_ID;

Para.Mode = EXTI11\_MODE;

Para.Port = EXTI11\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI12\_EN > 0)

Para.id = EXTI12\_ID;

Para.Mode = EXTI12\_MODE;

Para.Port = EXTI12\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI13\_EN > 0)

Para.id = EXTI13\_ID;

Para.Mode = EXTI13\_MODE;

Para.Port = EXTI13\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI14\_EN > 0)

Para.id = EXTI14\_ID;

Para.Mode = EXTI14\_MODE;

Para.Port = EXTI14\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (EXTI15\_EN > 0)

Para.id = EXTI15\_ID;

Para.Mode = EXTI15\_MODE;

Para.Port = EXTI15\_PORT;

EXTIX\_Init((EXTI\_PARA \*)&Para);

#endif

#if (DEBUG\_APP\_EN == 1)

printf("EXTI³õÊ¼»¯: OK\r\n");

#endif

}

void API\_Init(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

delay\_init(168);

#if (IWDG\_EN > 0)

IWDG\_Init();

IWDG\_Feed();

#endif

IO\_Init();

#if ((UART1\_EN > 0)||(UART2\_EN > 0)||(UART3\_EN > 0)||(UART4\_EN > 0)||(UART5\_EN > 0)||(UART6\_EN > 0))

Uart\_Init();

#endif

#if ((TIM2\_EN > 0)||(TIM3\_EN > 0)||(TIM4\_EN > 0)||(TIM5\_EN > 0)||(TIM6\_EN > 0)||(TIM7\_EN > 0))

TIM\_Init();

#endif

EXTI\_APPInit();

Logo\_Out();

#if (EEPROM\_EN > 0 )

EEPROM\_Init();

EEPROM\_Check();

if(EEPROM\_Check()==1)

{

#if (DEBUG\_APP\_EN == 1)

printf("EEPROM err");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("EEPROM ok");

#endif

}

#endif

#if (SPI1\_EN > 0 )

SPI1\_Init();

#endif

#if (SPIFLASH\_EN > 0 )

W25QXX\_Init();

#endif

#if (RTC\_EN > 0)

RTC\_APPInit();

#endif

}

#include "PIONEER\_Config.h"

#include <includes.h>

#include "STM32F4\_Head.h"

#include "UserVars.h"

#include "APP\_Init.h"

#include "OSVars.h"

#include "const.h"

static void App\_TaskCreate (void);

static void App\_TaskStart (void \*p\_arg);

#define VALIDDATA\_LEN 8

unsigned char buf[8]={0X01,0X03,0X00,0X00,0X00,0X01,0X84,0X0A};

#define RS485\_TASK\_PRIO 4

#define RS485\_STK\_SIZE 160

OS\_STK RS485\_TASK\_STK[RS485\_STK\_SIZE];

void rs485\_task(void \*pdata);

int main (void)

{

CPU\_INT08U os\_err;

API\_Init();

delay\_ms(10);

CPU\_IntDis();

OSInit();

EEPROM\_Init();

os\_err = OSTaskCreate((void (\*)(void \*)) App\_TaskStart,

(void \* ) 0,

(OS\_STK \* )&App\_TaskStartStk[APP\_TASK\_START\_STK\_SIZE - 1],

(INT8U ) APP\_TASK\_START\_PRIO

);

OSFlagSys = OSFlagCreate(0, (INT8U \*)&os\_err);

OSTimeSet(0);

#if (IWDG\_EN > 0)

IWDG\_Feed();

#endif

OSStart();

return (0);

}

static void App\_TaskStart (void \*p\_arg)

{

INT32U TaskCount;

(void)p\_arg;

#if (IWDG\_EN > 0)

IWDG\_Feed();

#endif

UserVars.Flag |= OS\_START\_FLAG;

#if (OS\_TASK\_STAT\_EN > 0)

OSStatInit();

#endif

App\_TaskCreate();

TaskCount = 0;

UserVars.TaskRunStatus = 0;

UserVars.TimerCount = 0;

while (1)

{

OSTimeDly(10);

#if (IWDG\_EN > 0)

IWDG\_Feed();

#endif

TaskCount++;

if ((TaskCount%10)==0)

{

UserVars.TimerCount++;

}

if (TaskCount>6000)

{

TaskCount = 0;

if (UserVars.TaskRunStatus == TASK\_FLAG)

{

UserVars.TaskRunStatus = 0;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskMain£ºÈÎÎñ³ö´í£¬³ö´í´úÂë£º%x\r\n\r\n",UserVars.TaskRunStatus);

#endif

}

}

}

}

static void App\_TaskCreate (void)

{

CPU\_INT08U os\_err;

INT8U err;

INT8U OS\_Err;

#if (TASK\_IO\_EN == 1)

os\_err = OSTaskCreate((void (\*)(void \*)) App\_TaskIO,

(void \* ) 0,

(OS\_STK \* )&App\_TaskIOStk[APP\_TASK\_IO\_STK\_SIZE - 1],

(INT8U ) APP\_TASK\_IO\_PRIO

);

if(OS\_ERR\_NONE == os\_err)

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat App\_TaskIO Success!\r\n");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat App\_TaskIO Error!\r\n");

#endif

}

#endif

#if (TASK\_TEST\_EN == 1)

os\_err = OSTaskCreate((void (\*)(void \*)) App\_TaskTest,

(void \* ) 0,

(OS\_STK \* )&App\_TaskTestStk[APP\_TASK\_TEST\_STK\_SIZE - 1],

(INT8U ) APP\_TASK\_TEST\_PRIO

);

if(OS\_ERR\_NONE == os\_err)

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat App\_TaskTest Success!\r\n");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat App\_TaskTest Error!\r\n");

#endif

}

#endif

#if (TASK\_NB\_EN == 1)

#if (NBTX\_TASK\_EN == 1)

os\_err = OSTaskCreate((void (\*)(void \*)) NB\_TxTask,

(void \* ) 0,

(OS\_STK \* )&NBTX\_TASK\_STK[NBTX\_TASK\_STK\_SIZE - 1],

(INT8U ) NBTX\_TASK\_PRIO

);

if(OS\_ERR\_NONE == os\_err)

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat NB\_TxTask Success!\r\n");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat NB\_TxTask Error!\r\n");

#endif

}

#endif

#if (NBRV\_TASK\_EN == 1)

os\_err = OSTaskCreate((void (\*)(void \*)) NB\_RvTask,

(void \* ) 0,

(OS\_STK \* )&NBRV\_TASK\_STK[NBRV\_TASK\_STK\_SIZE - 1],

(INT8U ) NBRX\_TASK\_PRIO

);

if(OS\_ERR\_NONE == os\_err)

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat NB\_RvTask Success!\r\n");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat NB\_RvTask Error!\r\n");

#endif

}

#endif

#if (UPDATE\_TASK\_EN == 1)

os\_err = OSTaskCreate((void (\*)(void \*)) NB\_UpdateTask,

(void \* ) 0,

(OS\_STK \* )&UPDATE\_TASK\_STK[UPDATE\_TASK\_STK\_SIZE - 1],

(INT8U ) UPDATE\_TASK\_PRIO

);

if(OS\_ERR\_NONE == os\_err)

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat NB\_UpdateTask Success!\r\n");

#endif

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat NB\_UpdateTask Error!\r\n");

#endif

}

#endif

#endif

#if (TASK\_UART\_EN == 1)

os\_err = OSTaskCreate((void (\*)(void \*)) App\_TaskUart,

(void \* ) 0,

(OS\_STK \* )&App\_TaskUartStk[APP\_TASK\_UART\_STK\_SIZE - 1],

(INT8U ) APP\_TASK\_UART\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat App\_TaskUart!\r\n");

#endif

#endif

#if (TASK\_MLX\_EN == 1)

os\_err = OSTaskCreate(

(void(\*)(void \*))mlx\_task,

(void \*)0,

(OS\_STK \*)&MLX\_TASK\_STK[MLX\_STK\_SIZE-1],

(INT8U )MLX\_TASK\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat MLX Success!\r\n");

#endif

#endif

#if (TASK\_SOSKEY\_EN == 1)

os\_err = OSTaskCreate(

(void(\*)(void \*))soskey\_task,

(void \*)0,

(OS\_STK \*)&SOSKEY\_TASK\_STK[SOSKEY\_STK\_SIZE-1],

(INT8U )SOSKEY\_TASK\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat soskey Success!\r\n");

#endif

#endif

#if (TASK\_BJ\_EN == 1)

os\_err = OSTaskCreate(

(void(\*)(void \*))bj\_task,

(void \*)0,

(OS\_STK \*)&BJ\_TASK\_STK[BJ\_STK\_SIZE],

(INT8U )BJ\_TASK\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat bj Success!\r\n");

#endif

#endif

#if (TASK\_AO\_EN == 1)

os\_err = OSTaskCreate(

(void(\*)(void \*))ao\_task,

(void \*)0,

(OS\_STK \*)&AO\_TASK\_STK[AO\_STK\_SIZE],

(INT8U )AO\_TASK\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat ao Success!\r\n");

#endif

#endif

#if (TASK\_ACC\_EN == 1)

os\_err = OSTaskCreate(

(void(\*)(void \*))acc\_task,

(void \*)0,

(OS\_STK \*)&ACC\_TASK\_STK[ACC\_STK\_SIZE],

(INT8U )ACC\_TASK\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat ao Success!\r\n");

#endif

#endif

#if (TASK\_BD\_EN == 1)

os\_err = OSTaskCreate(

(void(\*)(void \*))bd\_task,

(void \*)0,

(OS\_STK \*)&BD\_TASK\_STK[BD\_STK\_SIZE],

(INT8U )BD\_TASK\_PRIO

);

#if (DEBUG\_APP\_EN == 1)

printf("TaskCreate£ºCreat ao Success!\r\n");

#endif

#endif

NBUartRdy = OSSemCreate(0);

NBDataQ = OSQCreate(NBData, 32);

NBDataMem = OSMemCreate((void\*)NBMemData,16,256,&err);

NBStateSem = OSSemCreate(0);

NBUpdateSem = OSSemCreate(0);

NBUartMutex = OSMutexCreate(4,&OS\_Err);

Uart6Mutex = OSMutexCreate(5,&OS\_Err);

BJSTARTSem=OSSemCreate(0);

BJOUTSem=OSSemCreate(0);

}

#include "UserVars.h"

USER\_VARS UserVars;

void \*DisplayQMsg[32];

INT8U DisMemData[32][64];

INT8U UART1\_RX\_BUF[UART1\_RXBUF\_SIZE];

INT8U UART2\_RX\_BUF[UART2\_RXBUF\_SIZE];

INT8U UART3\_RX\_BUF[UART3\_RXBUF\_SIZE];

INT8U UART4\_RX\_BUF[UART4\_RXBUF\_SIZE];

INT8U UART5\_RX\_BUF[UART5\_RXBUF\_SIZE];

INT8U UART6\_RX\_BUF[UART6\_RXBUF\_SIZE];

INT16U UART1\_RX\_STA=0;

INT16U UART2\_RX\_STA=0;

INT16U UART3\_RX\_STA=0;

INT16U UART4\_RX\_STA=0;

INT16U UART5\_RX\_STA=0;

INT16U UART6\_RX\_STA=0;

INT8U TIM4CH1\_CAPTURE\_STA=0;

INT32U TIM4CH1\_CAPTURE\_VAL;

INT8U ConnectStatus=0;

INT8U DelCnt=0;

INT8U NBRvDelay=0;

INT16U NBUpdateCnt;

INT8U SendIntervalSecond=3;

INT16U SendIntervalMilli=0;

INT8U ReadSIData[6];

INT8U HaveOk=0;

#define CONFIG\_PARA\_LEN sizeof(CONFIG\_PARA)

#define TASK\_IO\_FLAG 0x00000001

#define TASK\_TEST\_FLAG 0x00000002

#define NBTX\_TASK\_FLAG 0x00000004

#define NBRV\_TASK\_FLAG 0x00000008

#define UPDATE\_TASK\_FLAG 0x00000010

#define TASK\_UART\_FLAG 0x00000020

#define TASK\_BJ\_FLAG 0x00000040

#define TASK\_AO\_FLAG 0x00000080

#define TASK\_MLX\_FLAG 0x00000100

#define TASK\_SOSKEY\_FLAG 0x00000200

#define TASK\_ACC\_FLAG 0x00000400

#define TASK\_BD\_FLAG 0x00000800

#define TASK\_FLAG (TASK\_TEST\_FLAG\*TASK\_TEST\_EN + TASK\_IO\_FLAG\*TASK\_IO\_EN + NBTX\_TASK\_FLAG\*NBTX\_TASK\_EN + \

NBRV\_TASK\_FLAG\*NBRV\_TASK\_EN + UPDATE\_TASK\_FLAG\*UPDATE\_TASK\_EN + TASK\_BJ\_FLAG\*TASK\_BJ\_EN +TASK\_BD\_FLAG\*TASK\_BD\_EN\

+TASK\_AO\_FLAG\*TASK\_AO\_EN+TASK\_MLX\_FLAG\*TASK\_MLX\_EN+TASK\_SOSKEY\_FLAG\*TASK\_SOSKEY\_EN+TASK\_ACC\_FLAG\*TASK\_ACC\_EN )

#define RTC\_SECIT\_FLAG 0x00000001

#define RTC\_ALRIT\_FLAG 0x00000002

#define SPIFLASH\_WK\_FLAG 0x00000004

#define SD\_WK\_FLAG 0x00000008

#define OS\_START\_FLAG 0x00000020

#define SPIFLASH\_OK\_FLAG 0x00000001

#define NFLASH\_OK\_FLAG 0x00000002

#define EEPROM\_OK\_FLAG 0x00000004

#define SRAM\_OK\_FLAG 0x00000008

#define SD\_OK\_FLAG 0x00000010

#define DATABUF\_SIZE 512

#pragma pack (1)

typedef struct

{

INT32U Flag;

INT32U TaskRunStatus;

INT32U OKFlag;

INT32U DI;

INT32U DO;

INT32U TimerCount;

INT8U buf[DATABUF\_SIZE+1];

INT16U UartData;

}USER\_VARS;

#pragma pack()

extern USER\_VARS UserVars;

extern INT8U UART1\_RX\_BUF[UART1\_RXBUF\_SIZE];

extern INT8U UART2\_RX\_BUF[UART2\_RXBUF\_SIZE];

extern INT8U UART3\_RX\_BUF[UART3\_RXBUF\_SIZE];

extern INT8U UART4\_RX\_BUF[UART4\_RXBUF\_SIZE];

extern INT8U UART5\_RX\_BUF[UART5\_RXBUF\_SIZE];

extern INT8U UART6\_RX\_BUF[UART6\_RXBUF\_SIZE];

extern INT16U UART1\_RX\_STA;

extern INT16U UART2\_RX\_STA;

extern INT16U UART3\_RX\_STA;

extern INT16U UART4\_RX\_STA;

extern INT16U UART5\_RX\_STA;

extern INT16U UART6\_RX\_STA;

extern INT8U TIM4CH1\_CAPTURE\_STA;

extern INT32U TIM4CH1\_CAPTURE\_VAL;

extern INT8U ConnectStatus;

extern INT8U DelCnt;

extern INT8U NBRvDelay;

extern INT16U NBUpdateCnt;

extern INT8U SendIntervalSecond;

extern INT16U SendIntervalMilli;

extern INT8U ReadSIData[6];

extern INT8U HaveOk;

typedef struct

{

char longitude1[9];

char latitude[9];

char sos;

char tem;

char pressure;

char accex;

char accey;

char accez;

}threshhold;

extern threshhold safe;

#endif

#include "includes.h"

#include "NB\_IoT.h"

#include "STM32F4\_Head.h"

#include "UserVars.h"

#include "OSVars.h"

#include "string.h"

#include "stdio.h"

#define RV\_BUF\_LEN 256

INT8U RvBuf[RV\_BUF\_LEN] = {0};

char testtmp=0x30;

INT8U Close\_Date[]={"AT+MIPLCLOSE=0\r\n"};

INT8U Create\_Date[]={"AT+MIPLCREATE=49,130031F10003F2002304001100000000000000123138332E3233302E34302E34303A35363833000131F30008C000000000,0,49,0\r\n"};

INT8U AddObj\_Date[]={"AT+MIPLADDOBJ=0,3200,1,\"1\",1,0\r\n"};

INT8U DiscoverRsp\_Date[]={"AT+MIPLDISCOVERRSP=0,3200,1,4,\"5750\"\r\n"};

INT8U Open\_Date[]={"AT+MIPLOPEN=0,3000,30\r\n"};

INT8U Del\_Data[]={"AT+MIPLDELETE=0\r\n"};

INT8U Query\_Data[]={"AT+MIPLOPEN?\r\n"};

INT8U NRB\_Data[]={"AT+NRB\r\n"};

INT8U Update\_Data[]={"AT+MIPLUPDATE=0,0,0\r\n"};

INT8U nbiot\_Create(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 0;

Uart\_Write(UART5\_ID,Create\_Date,sizeof(Create\_Date));

#if (NBRV\_TASK\_EN==1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

if((UART5\_RX\_STA&0x8000))

{

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("CreateÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

if((RvBuf[19] == 'O') && (RvBuf[20] == 'K'))

{

printf("´´½¨Éè±¸ÊµÌå³É¹¦\r\n");

result = 1;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("´´½¨Éè±¸ÊµÌåÊ§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_AddObj(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 0;

Uart\_Write(UART5\_ID,AddObj\_Date,sizeof(AddObj\_Date));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("AddobjÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

if((RvBuf[2] == 'O') && (RvBuf[3] == 'K'))

{

printf("¶©ÔÄOBJ×éÅäÖÃ³É¹¦\r\n");

result = 1;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("¶©ÔÄOBJ×éÅäÖÃÊ§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_DiscoverRsp(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 0;

Uart\_Write(UART5\_ID,DiscoverRsp\_Date,sizeof(DiscoverRsp\_Date));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("DiscoverRspÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

if((RvBuf[2] == 'O') && (RvBuf[3] == 'K'))

{

#if (DEBUG\_APP\_EN == 1)

printf("¶©ÔÄResource²ÎÊýÅäÖÃ³É¹¦\r\n");

#endif

result = 1;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("¶©ÔÄResource²ÎÊýÅäÖÃÊ§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_Connect(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 1;

Uart\_Write(UART5\_ID,Open\_Date,sizeof(Open\_Date));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("ConnectÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

for(i=0; i<RV\_BUF\_LEN; i++)

{

if(RvBuf[i] == 0x36)

{

#if (DEBUG\_APP\_EN == 1)

printf("ÏòÆ½Ì¨×¢²áµÇÂ¼³É¹¦\r\n");

#endif

result = 1;

break;

}

}

if(i >= RV\_BUF\_LEN)

{

#if (DEBUG\_APP\_EN == 1)

printf("ÏòÆ½Ì¨×¢²áµÇÂ¼Ê§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_Query(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 1;

Uart\_Write(UART5\_ID,Query\_Data,sizeof(Query\_Data));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("QueryÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

for(i=0; i<RV\_BUF\_LEN; i++)

{

if(RvBuf[i] == 0x36)

{

#if (DEBUG\_APP\_EN == 1)

printf("Ä£×éÒÑµÇÂ¼OneNet\r\n");

#endif

result = 1;

break;

}

}

if(i >= RV\_BUF\_LEN)

{

#if (DEBUG\_APP\_EN == 1)

printf("Ä£×éÎ´µÇÂ¼OneNet\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_Update(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 1;

Uart\_Write(UART5\_ID,Update\_Data,sizeof(Update\_Data));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("nbiot\_UpdateµÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

for(i=0; i<RV\_BUF\_LEN-1; i++)

{

if((RvBuf[i] == 0x31) && (RvBuf[i+1] == 0x31))

{

#if (DEBUG\_APP\_EN == 1)

printf("´æ»îÊ±¼ä¸üÐÂ³É¹¦\r\n");

#endif

result = 1;

break;

}

}

if(i >= RV\_BUF\_LEN)

{

#if (DEBUG\_APP\_EN == 1)

printf("´æ»îÊ±¼ä¸üÐÂÊ§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_Nofity(INT8U \*NBBuff)

{

INT8U flag=0;

INT16U i,j;

INT8U OS\_Err;

INT8U Notify\_Date[128]={0};

sprintf((char\*)Notify\_Date,"AT+MIPLNOTIFY=0,0,3200,0,5750,1,%d,\"%s\",0,0\r\n",strlen((char\*)NBBuff),NBBuff);

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 0;

Uart\_Write(UART5\_ID,Notify\_Date,sizeof(Notify\_Date));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if(0 == HaveOk)

{

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("NofityÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

if((RvBuf[2] == 'O') && (RvBuf[3] == 'K'))

{

#if (DEBUG\_APP\_EN == 1)

printf("ÉÏ´«Êý¾Ý³É¹¦\r\n");

#endif

flag = 1;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("ÉÏ´«Êý¾ÝÊ§°Ü\r\n");

#endif

flag = 0;

}

}

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("ÉÏ´«Êý¾Ý³É¹¦\r\n");

#endif

HaveOk = 0;

OSMutexPost(NBUartMutex);

flag = 1;

}

return flag;

}

INT8U nbiot\_Del(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 0;

Uart\_Write(UART5\_ID,Del\_Data,sizeof(Del\_Data));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err);

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("DelÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

if((RvBuf[2] == 'O') && (RvBuf[3] == 'K'))

{

#if (DEBUG\_APP\_EN == 1)

printf("É¾³ýÉè±¸ÊµÌå³É¹¦\r\n");

#endif

result = 1;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("É¾³ýÉè±¸ÊµÌåÊ§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

INT8U nbiot\_Reset(void)

{

INT16U i,j;

INT8U result;

INT8U OS\_Err;

OSMutexPend(NBUartMutex, 0, &OS\_Err);

NBRvDelay = 0;

Uart\_Write(UART5\_ID,NRB\_Data,sizeof(NRB\_Data));

#if (NBRV\_TASK\_EN == 1)

OSSemPend(NBStateSem, 500, &OS\_Err );

#else

OSSemPend(NBUartRdy, 500, &OS\_Err);

#endif

if((UART5\_RX\_STA&0x8000))

{

for(i=0; i<RV\_BUF\_LEN; i++)

{

RvBuf[i] = 0;

}

for(j=0; j<(UART5\_RX\_STA&0X3FFF); j++)

{

RvBuf[j] = UART5\_RX\_BUF[j];

UART5\_RX\_BUF[j] = 0;

}

UART5\_RX\_STA=0;

OSMutexPost(NBUartMutex);

#if (DEBUG\_APP\_EN == 1)

printf("ResetÊÕµ½µÄÊý¾ÝÊÇ%s\r\n", RvBuf);

#endif

if((RvBuf[2] == 'R') && (RvBuf[3] == 'E'))

{

#if (DEBUG\_APP\_EN == 1)

printf("NBÄ£×é¸´Î»³É¹¦\r\n");

#endif

result = 1;

}

else

{

#if (DEBUG\_APP\_EN == 1)

printf("NBÄ£×é¸´Î»Ê§°Ü\r\n");

#endif

result = 0;

}

}

return result;

}

#include "sys.h"

#include "STM32F4\_Head.h"

#include "PIONEER\_Config.h"

#include "const.h"

#include "includes.h"

#include "UserVars.h"

#include "OSVars.h"

void EXTI0\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line0) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI0²úÉúÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line0);

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void EXTI1\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line1) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI1²úÉúÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line1);

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void EXTI2\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line2) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI2²úÉúÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line2);

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void EXTI3\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line3) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI3²úÉúÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line3);

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void EXTI4\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line4) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI4²úÉúÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line4);

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void EXTI9\_5\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line5) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI5²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line5);

}

if(EXTI\_GetITStatus(EXTI\_Line6) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI6²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line6);

}

if(EXTI\_GetITStatus(EXTI\_Line7) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI7²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line7);

}

if(EXTI\_GetITStatus(EXTI\_Line8) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI8²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line8);

}

if(EXTI\_GetITStatus(EXTI\_Line9) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI9²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line9);

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void EXTI15\_10\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

delay\_ms(10);

if(EXTI\_GetITStatus(EXTI\_Line10) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI10²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line10);

}

if(EXTI\_GetITStatus(EXTI\_Line11) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI11²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line11);

}

if(EXTI\_GetITStatus(EXTI\_Line12) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI12²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line12);

}

if(EXTI\_GetITStatus(EXTI\_Line13) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI13²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line13);

}

if(EXTI\_GetITStatus(EXTI\_Line14) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI14²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line14);

}

if(EXTI\_GetITStatus(EXTI\_Line15) != RESET)

{

#if (DEBUG\_APP\_EN > 0)

printf("EXTI15²úÉúÖÐ¶Ï\r\n");

#endif

EXTI\_ClearITPendingBit(EXTI\_Line15);

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM2\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM2,TIM\_IT\_Update)==SET)

{

#if (UART1\_EN>0 && UART1\_RX\_MODE==0)

UART1\_RX\_STA|=1<<15;

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update);

TIM\_Cmd(TIM2, DISABLE);

#else

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM3\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM3,TIM\_IT\_Update)==SET)

{

#if (UART2\_EN>0 && UART2\_RX\_MODE==0)

UART2\_RX\_STA|=1<<15;

TIM\_ClearITPendingBit(TIM3, TIM\_IT\_Update);

TIM\_Cmd(TIM3, DISABLE);

#else

NBUpdateCnt++;

if(NBUpdateCnt >= UPDATE\_TIME\_DATA\*60)

{

TIM\_ClearITPendingBit(TIM3, TIM\_IT\_Update );

TIM\_Cmd(TIM3, DISABLE);

OSSemPost(NBUpdateSem);

}

TIM\_ClearITPendingBit(TIM3, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM4\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM4,TIM\_IT\_Update)==SET)

{

#if (UART3\_EN>0 && UART3\_RX\_MODE==0)

UART3\_RX\_STA|=1<<15;

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update);

TIM\_Cmd(TIM4, DISABLE);

#else

NBUpdateCnt++;

if(NBUpdateCnt >= UPDATE\_TIME\_DATA\*60)

{

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update );

TIM\_Cmd(TIM4, DISABLE);

OSSemPost(NBUpdateSem);

}

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM4\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if((TIM4CH1\_CAPTURE\_STA&0X80)==0)

{

if(TIM\_GetITStatus(TIM4, TIM\_IT\_Update) != RESET)

{

if(TIM4CH1\_CAPTURE\_STA&0X40)

{

if((TIM4CH1\_CAPTURE\_STA&0X3F)==0X3F)

{

TIM4CH1\_CAPTURE\_STA|=0X80;

TIM4CH1\_CAPTURE\_VAL=0XFFFF;

}else TIM4CH1\_CAPTURE\_STA++;

}

}

if(TIM\_GetITStatus(TIM4, TIM\_IT\_CC1) != RESET)

{

if(TIM4CH1\_CAPTURE\_STA&0X40)

{

TIM4CH1\_CAPTURE\_STA|=0X80;

TIM4CH1\_CAPTURE\_VAL=TIM\_GetCapture1(TIM4);

TIM\_OC1PolarityConfig(TIM4,TIM\_ICPolarity\_Rising);

}else

{

TIM4CH1\_CAPTURE\_STA=0;

TIM4CH1\_CAPTURE\_VAL=0;

TIM\_SetCounter(TIM4,0);

TIM4CH1\_CAPTURE\_STA|=0X40;

TIM\_OC1PolarityConfig(TIM4,TIM\_ICPolarity\_Falling);

}

}

}

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_CC1|TIM\_IT\_Update);

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM4\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM4,TIM\_IT\_Update)==SET)

{

#if (UART3\_EN>0 && UART3\_RX\_MODE==0)

UART3\_RX\_STA|=1<<15;

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update);

TIM\_Cmd(TIM4, DISABLE);

#else

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM5\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM5,TIM\_IT\_Update)==SET)

{

#if (UART5\_EN>0 && UART5\_RX\_MODE==0)

if(0 == NBRvDelay)

{

UART5\_RX\_STA|=1<<15;

OSSemPost(NBUartRdy);

OSSemPost(FurnaceRdy);

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

TIM\_Cmd(TIM5, DISABLE);

}

else if (NBRvDelay >= 10)

{

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

NBRvDelay = 0;

}

else

{

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

NBRvDelay++;

}

#else

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

/\*#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM5,TIM\_IT\_Update)==SET)

{

#if (UART4\_EN>0 && UART4\_RX\_MODE==0)

UART4\_RX\_STA|=1<<15;

OSSemPost(ComSem);

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

TIM\_Cmd(TIM5, DISABLE);

#else

TIM\_ClearITPendingBit(TIM5, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif\*/

}

void TIM6\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM6,TIM\_IT\_Update)==SET)

{

#if (UART5\_EN>0 && UART5\_RX\_MODE==0)

if(0 == NBRvDelay)

{

UART5\_RX\_STA|=1<<15;

OSSemPost(NBUartRdy);

TIM\_ClearITPendingBit(TIM6, TIM\_IT\_Update);

TIM\_Cmd(TIM6, DISABLE);

}

else if (NBRvDelay >= 10)

{

TIM\_ClearITPendingBit(TIM6, TIM\_IT\_Update);

NBRvDelay = 0;

}

else

{

TIM\_ClearITPendingBit(TIM6, TIM\_IT\_Update);

NBRvDelay++;

}

#else

TIM\_ClearITPendingBit(TIM6, TIM\_IT\_Update);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void TIM7\_IRQHandler(void)

{

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(TIM\_GetITStatus(TIM7,TIM\_IT\_Update)==SET)

{

#if (UART6\_EN>0 && UART6\_RX\_MODE==0)

UART6\_RX\_STA|=1<<15;

TIM\_ClearITPendingBit(TIM7, TIM\_IT\_Update);

TIM\_Cmd(TIM7, DISABLE);

#else

TIM\_ClearITPendingBit(TIM7, TIM\_IT\_Update);

TIM\_Cmd(TIM7, DISABLE);

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void RTC\_Alarm\_IRQHandler(void)

{

if(RTC\_GetFlagStatus(RTC\_FLAG\_ALRAF)==SET)

{

RTC\_ClearFlag(RTC\_FLAG\_ALRAF);

#if (DEBUG\_APP\_EN == 1)

printf("RTC²úÉúÄÖÖÓ±¨¾¯ÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line17);

}

void RTC\_WKUP\_IRQHandler(void)

{

if(RTC\_GetFlagStatus(RTC\_FLAG\_WUTF)==SET)

{

RTC\_ClearFlag(RTC\_FLAG\_WUTF);

#if (DEBUG\_APP\_EN == 1)

printf("RTC²úÉú»½ÐÑÖÐ¶Ï\r\n");

#endif

}

EXTI\_ClearITPendingBit(EXTI\_Line22);

}

void USART1\_IRQHandler(void)

{

u8 Res;

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET)

{

Res =USART\_ReceiveData(USART1);

#if (UART1\_RX\_MODE == 1)

if((UART1\_RX\_STA&0x8000)==0)

{

if(UART1\_RX\_STA&0x4000)

{

if(Res!=0x0a)

UART1\_RX\_STA=0;

else

UART1\_RX\_STA|=0x8000;

}

else

{

if(Res==0x0d)

UART1\_RX\_STA|=0x4000;

else

{

UART1\_RX\_BUF[UART1\_RX\_STA&0X3FFF]=Res ;

UART1\_RX\_STA++;

if(UART1\_RX\_STA>(UART1\_RXBUF\_SIZE-1))

UART1\_RX\_STA=0;

}

}

}

#elif (UART1\_RX\_MODE == 2)

if(UART1\_RX\_STA<UART1\_RXBUF\_SIZE)

{

UART1\_RX\_BUF[UART1\_RX\_STA]=Res;

UART1\_RX\_STA++;

}

#else

if((UART1\_RX\_STA&(1<<15))==0)

{

if(UART1\_RX\_STA<UART1\_RXBUF\_SIZE)

{

TIM\_SetCounter(TIM2,0);

if(UART1\_RX\_STA==0)

{

TIM\_Cmd(TIM2, ENABLE);

}

UART1\_RX\_BUF[UART1\_RX\_STA++]=Res;

}

else

{

UART1\_RX\_STA|=1<<15;

}

}

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void USART2\_IRQHandler(void)

{

u8 Res;

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(USART\_GetITStatus(USART2, USART\_IT\_RXNE) != RESET)

{

Res =USART\_ReceiveData(USART2);

#if (UART2\_RX\_MODE == 1)

if((UART2\_RX\_STA&0x8000)==0)

{

if(UART2\_RX\_STA&0x4000)

{

if(Res!=0x0a)

UART2\_RX\_STA=0;

else

UART2\_RX\_STA|=0x8000;

}

else

{

if(Res==0x0d)

UART2\_RX\_STA|=0x4000;

else

{

UART2\_RX\_BUF[UART2\_RX\_STA&0X3FFF]=Res ;

UART2\_RX\_STA++;

if(UART2\_RX\_STA>(UART2\_RXBUF\_SIZE-1))

UART2\_RX\_STA=0;

}

}

}

#elif (UART2\_RX\_MODE == 2)

if(UART2\_RX\_STA<UART2\_RXBUF\_SIZE)

{

UART2\_RX\_BUF[UART2\_RX\_STA]=Res;

UART2\_RX\_STA++;

}

#else

if((UART2\_RX\_STA&(1<<15))==0)

{

if(UART2\_RX\_STA<UART2\_RXBUF\_SIZE)

{

TIM\_SetCounter(TIM3,0);

if(UART2\_RX\_STA==0)

{

TIM\_Cmd(TIM3, ENABLE);

}

UART2\_RX\_BUF[UART2\_RX\_STA++]=Res;

}

else

{

UART2\_RX\_STA|=1<<15;

}

}

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void USART3\_IRQHandler(void)

{

u8 Res;

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(USART\_GetITStatus(USART3, USART\_IT\_RXNE) != RESET)

{

Res =USART\_ReceiveData(USART3);

#if (UART3\_RX\_MODE == 1)

if((UART3\_RX\_STA&0x8000)==0)

{

if(UART3\_RX\_STA&0x4000)

{

if(Res!=0x0a)

UART3\_RX\_STA=0;

else

UART3\_RX\_STA|=0x8000;

}

else

{

if(Res==0x0d)

UART3\_RX\_STA|=0x4000;

else

{

UART3\_RX\_BUF[UART3\_RX\_STA&0X3FFF]=Res ;

UART3\_RX\_STA++;

if(UART3\_RX\_STA>(UART3\_RXBUF\_SIZE-1))

UART3\_RX\_STA=0;

}

}

}

#elif (UART3\_RX\_MODE == 2)

if(UART3\_RX\_STA<UART3\_RXBUF\_SIZE)

{

UART3\_RX\_BUF[UART3\_RX\_STA]=Res;

UART3\_RX\_STA++;

}

#else

if((UART3\_RX\_STA&(1<<15))==0)

{

if(UART3\_RX\_STA<UART3\_RXBUF\_SIZE)

{

TIM\_SetCounter(TIM4,0);

if(UART3\_RX\_STA==0)

{

TIM\_Cmd(TIM4, ENABLE);

}

UART3\_RX\_BUF[UART3\_RX\_STA++]=Res;

}

else

{

UART3\_RX\_STA|=1<<15;

}

}

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void UART4\_IRQHandler(void)

{

u8 Res;

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(USART\_GetITStatus(UART4, USART\_IT\_RXNE) != RESET)

{

Res =USART\_ReceiveData(UART4);

#if (UART4\_RX\_MODE == 1)

if((UART4\_RX\_STA&0x8000)==0)

{

if(UART4\_RX\_STA&0x4000)

{

if(Res!=0x0a)

UART4\_RX\_STA=0;

else

UART4\_RX\_STA|=0x8000;

}

else

{

if(Res==0x0d)

UART4\_RX\_STA|=0x4000;

else

{

UART4\_RX\_BUF[UART4\_RX\_STA&0X3FFF]=Res ;

UART4\_RX\_STA++;

if(UART4\_RX\_STA>(UART4\_RXBUF\_SIZE-1))

UART4\_RX\_STA=0;

}

}

}

#elif (UART4\_RX\_MODE == 2)

if(UART4\_RX\_STA<UART4\_RXBUF\_SIZE)

{

UART4\_RX\_BUF[UART4\_RX\_STA]=Res;

UART4\_RX\_STA++;

}

#else

if((UART4\_RX\_STA&(1<<15))==0)

{

if(UART4\_RX\_STA<UART4\_RXBUF\_SIZE)

{

TIM\_SetCounter(TIM5,0);

if(UART4\_RX\_STA==0)

{

TIM\_Cmd(TIM5, ENABLE);

}

UART4\_RX\_BUF[UART4\_RX\_STA++]=Res;

}

else

{

UART4\_RX\_STA|=1<<15;

}

}

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void UART5\_IRQHandler(void)

{

u8 Res;

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(USART\_GetITStatus(UART5, USART\_IT\_RXNE) != RESET)

{

Res =USART\_ReceiveData(UART5);

#if (UART5\_RX\_MODE == 1)

if((UART5\_RX\_STA&0x8000)==0)

{

if(UART5\_RX\_STA&0x4000)

{

if(Res!=0x0a)

UART5\_RX\_STA=0;

else

UART5\_RX\_STA|=0x8000;

}

else

{

if(Res==0x0d)

UART5\_RX\_STA|=0x4000;

else

{

UART5\_RX\_BUF[UART5\_RX\_STA&0X3FFF]=Res ;

UART5\_RX\_STA++;

if(UART5\_RX\_STA>(UART5\_RXBUF\_SIZE-1))

UART5\_RX\_STA=0;

}

}

}

#elif (UART5\_RX\_MODE == 2)

if(UART5\_RX\_STA<UART5\_RXBUF\_SIZE)

{

UART5\_RX\_BUF[UART5\_RX\_STA]=Res;

UART5\_RX\_STA++;

}

#else

if((UART5\_RX\_STA&(1<<15))==0)

{

if(UART5\_RX\_STA<UART5\_RXBUF\_SIZE)

{

TIM\_SetCounter(TIM5,0);

if(UART5\_RX\_STA==0)

{

TIM\_Cmd(TIM5, ENABLE);

}

UART5\_RX\_BUF[UART5\_RX\_STA++]=Res;

}

else

{

UART5\_RX\_STA|=1<<15;

}

}

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

void USART6\_IRQHandler(void)

{

u8 Res;

#if UCOS\_II\_EN

OSIntEnter();

#endif

if(USART\_GetITStatus(USART6, USART\_IT\_RXNE) != RESET)

{

Res =USART\_ReceiveData(USART6);

if(UART6\_RX\_STA<64)

{

UART6\_RX\_BUF[UART6\_RX\_STA]=Res;

UART6\_RX\_STA++;

}

#if (UART6\_RX\_MODE == 1)

if((UART6\_RX\_STA&0x8000)==0)

{

if(UART6\_RX\_STA&0x4000)

{

if(Res!=0x0a)

UART6\_RX\_STA=0;

else

UART6\_RX\_STA|=0x8000;

}

else

{

if(Res==0x0d)

UART6\_RX\_STA|=0x4000;

else

{

UART6\_RX\_BUF[UART6\_RX\_STA&0X3FFF]=Res ;

UART6\_RX\_STA++;

if(UART6\_RX\_STA>(UART6\_RXBUF\_SIZE-1))

UART6\_RX\_STA=0;

}

}

}

#elif (UART6\_RX\_MODE == 2)

if(UART6\_RX\_STA<UART6\_RXBUF\_SIZE)

{

UART6\_RX\_BUF[UART6\_RX\_STA]=Res;

UART6\_RX\_STA++;

}

#else

if((UART6\_RX\_STA&(1<<15))==0)

{

if(UART6\_RX\_STA<UART6\_RXBUF\_SIZE)

{

TIM\_SetCounter(TIM7,0);

if(UART6\_RX\_STA==0)

{

TIM\_Cmd(TIM7, ENABLE);

}

UART6\_RX\_BUF[UART6\_RX\_STA++]=Res;

}

else

{

UART6\_RX\_STA|=1<<15;

}

}

#endif

}

#if UCOS\_II\_EN

OSIntExit();

#endif

}

#include "PIONEER\_Config.h"

#if (TASK\_NB\_EN == 1)

#include "STM32F4\_Head.h"

#include "UserVars.h"

#include "OSVars.h"

#include "NB\_IoT.h"

#include "stdio.h"

#include "string.h"

#define RS485BUFF\_LEN 46

#define VALIDDATA\_LEN 150

INT8U ReadErrInfo[]="ERR01";

#if (NBTX\_TASK\_EN == 1)

void NB\_TxTask(void \*pdata)

{

INT8U \*Msg,\*MemPtr;

INT8U OS\_Err;

INT16U i;

INT8U GetData[VALIDDATA\_LEN];

INT8U NB\_flag,send\_flag;

INT8U Create\_flag,Obj\_flag,Rsp\_flag,connect\_flag;

volatile u32 \*Num0;

threshhold safe;

double lati=110.23;

double longi=50.36;

for(i=0; i<VALIDDATA\_LEN; i++)

{

GetData[i] = 0;

}

ConnectStatus = 0;

for(i=0;i<9;i++)

{

lati+=0.5;

longi+=0.5;

safe.latitude[i]=(char)lati;

safe.longitude1[i]=(char)longi;

}

while(1)

{

Num0 = OSMemGet(NBDataMem, &OS\_Err);

if(OS\_Err == OS\_ERR\_NONE)

{

sprintf((char \*)Num0,"E,%s,%s,%s,%s,%s,%s,%s,%s",safe.latitude,safe.longitude1,safe.sos,safe.tem,safe.pressure,safe.accex,safe.accey,safe.accez);

OSQPost(NBDataQ, (void \*)Num0);

}

printf("nb\_tx runing!\r\n");

UserVars.TaskRunStatus |= NBTX\_TASK\_FLAG;

Msg = (INT8U\*)OSQPend(NBDataQ,0,&OS\_Err);

if(OS\_ERR\_NONE == OS\_Err)

{

memset(GetData,0,VALIDDATA\_LEN);

if(\*Msg != 'E')

{

MemPtr = Msg;

for(i=0; i<VALIDDATA\_LEN; i++)

{

GetData[i] = \*Msg;

Msg++;

}

OSMemPut(NBDataMem,(void\*)MemPtr);

}

else

{

MemPtr = Msg;

for(i=0; i<strlen((char \*)ReadErrInfo); i++)

{

GetData[i] = \*Msg;

Msg++;

}

OSMemPut(NBDataMem,(void\*)MemPtr);

}

NB\_flag = 0;

send\_flag = 0;

DelCnt = 0;

while(!NB\_flag)

{

if(1 == ConnectStatus)

{

send\_flag = nbiot\_Nofity((INT8U\*)GetData);

if(1 == send\_flag)

{

NB\_flag = 1;

DelCnt = 0;

break;

}

else

{

nbiot\_Del();

ConnectStatus = 0;

NB\_flag = 0;

DelCnt++;

}

}

if(0 == ConnectStatus)

{

Create\_flag = 0;

Obj\_flag = 0;

connect\_flag = 0;

Rsp\_flag = 0;

if(nbiot\_Query())

{

printf("C\r\n");

ConnectStatus = 1;

}

else

{

if(nbiot\_Create())

{

Create\_flag = 1;

}

if(Create\_flag == 1)

{

if(nbiot\_AddObj())

{

Obj\_flag = 1;

}

}

if((Create\_flag == 1) &&(Obj\_flag == 1))

{

if(nbiot\_DiscoverRsp())

{

Rsp\_flag = 1;

}

}

if((Create\_flag == 1) && (Obj\_flag ==1) && (Rsp\_flag == 1))

{

if(nbiot\_Connect())

{

connect\_flag = 1;

ConnectStatus = 1;

}

}

if((Create\_flag == 0) || (Obj\_flag == 0) || (connect\_flag == 0) || (Rsp\_flag == 0))

{

nbiot\_Del();

ConnectStatus = 0;

DelCnt++;

}

if(DelCnt >= 3)

{

nbiot\_Reset();

OSTaskSuspend(APP\_TASK\_START\_PRIO);

OSTaskSuspend(NBTX\_TASK\_PRIO);

}

}

}

}

}

}

}

#endif

#if (NBRV\_TASK\_EN == 1)

#define SIDataAdress 0

void NB\_RvTask(void \*pdata)

{

INT8U OS\_Err,i,len,Num;

INT8U NBRvData[150];

INT8U ResposeData[32];

INT8U msgid[10]={0};

INT8U StartNum;

INT8U DataLen;

INT8U SIData[100];

INT8U RunCnt;

INT8U PlusPosi;

while(1)

{

printf("nb\_rv runing!\r\n");

UserVars.TaskRunStatus |= NBRV\_TASK\_FLAG;

OSSemPend(NBUartRdy, 0, &OS\_Err);

len = (UART5\_RX\_STA&0X3FFF);

for(i=0; i<len; i++)

{

if(UART5\_RX\_BUF[i] == '+')

{

PlusPosi = i;

break;

}

}

if(i >= len)

{

OSSemPost(NBStateSem);

}

else

{

if((UART5\_RX\_BUF[PlusPosi] == '+') && (UART5\_RX\_BUF[PlusPosi+5] == 'W')

&& (UART5\_RX\_BUF[PlusPosi+6] == 'R') && (UART5\_RX\_BUF[PlusPosi+7] == 'I'))

{

HaveOk = 0;

for(i=0; i<len-1; i++)

{

if((UART5\_RX\_BUF[i] == 'O') &&(UART5\_RX\_BUF[i+1] == 'K'))

{

HaveOk = 1;

OSSemPost(NBStateSem);

break;

}

}

OSMutexPend(NBUartMutex, 0, &OS\_Err);

if((UART5\_RX\_STA&0x8000))

{

printf("%s",UART5\_RX\_BUF);

for(i=0; i<len; i++)

{

NBRvData[i] = UART5\_RX\_BUF[i];

UART5\_RX\_BUF[i] = 0;

}

UART5\_RX\_STA=0;

}

printf("%s",NBRvData);

for(i=0; i<10; i++)

{

msgid[i] = 0;

}

i= 0;

PlusPosi = PlusPosi + 13;

while(NBRvData[PlusPosi] != ',')

{

msgid[i] = NBRvData[PlusPosi];

PlusPosi++;

i++;

}

for(i=0; i<32; i++)

{

ResposeData[i] = 0;

}

sprintf((char\*)ResposeData,"AT+MIPLWRITERSP=0,%s,2\r\n",msgid);

RunCnt=0;

Num=0;

while((0 == RunCnt) && (Num < 3))

{

Uart\_Write(UART5\_ID,ResposeData,sizeof(ResposeData));

Num++;

OSSemPend(NBUartRdy, 0, &OS\_Err);

if((UART5\_RX\_STA&0x8000))

{

if((UART5\_RX\_BUF[2] == 'O') && (UART5\_RX\_BUF[3] == 'K'))

{

RunCnt = 1;

}

else

{

RunCnt = 0;

}

UART5\_RX\_STA=0;

}

}

OSMutexPost(NBUartMutex);

for(i=0;i<len; i++)

{

if((NBRvData[i] == 'F') && (NBRvData[i+1] == '2'))

StartNum=i;

}

for(i=StartNum;i<len; i++)

{

if(NBRvData[i] == ',' && NBRvData[i + 1] == '0')

DataLen= i- StartNum;

}

memcpy(SIData, &NBRvData[StartNum], DataLen);

printf("%s\r\n",SIData);

if((NBRvData[StartNum] == 'R') && (NBRvData[StartNum+1] == 'S'))

{

nbiot\_Reset();

nbiot\_Del();

OSTaskSuspend(APP\_TASK\_START\_PRIO);

OSTaskSuspend(NBTX\_TASK\_PRIO);

}

}

else

{

OSSemPost(NBStateSem);

}

}

}

}

#endif

#if (UPDATE\_TASK\_EN == 1)

void NB\_UpdateTask(void \*pdata)

{

u8 OS\_Err;

while(1)

{

printf("nb\_updata runing!\r\n");

UserVars.TaskRunStatus |= UPDATE\_TASK\_FLAG;

if(1 == ConnectStatus)

{

NBUpdateCnt = 0;

TIM\_Cmd(TIM3,ENABLE);

OSSemPend(NBUpdateSem, 0, &OS\_Err );

if(!nbiot\_Update())

{

if(nbiot\_Query())

{

if(!nbiot\_Update())

{

ConnectStatus = 0;

}

}

else

{

ConnectStatus = 0;

}

}

}

else

{

delay\_ms(1000);

}

}

}

#endif

#endif

#include "PIONEER\_Config.h"

#if (TASK\_SOSKEY\_EN == 1)

#include "STM32F4\_Head.h"

#include "OSVars.h"

#include "UserVars.h"

#include "string.h"

void soskey\_task(void \*pdata)

{

INT8U OS\_Err;

threshhold safe;

volatile u32 \*Num0;

pdata=pdata;

while(1)

{

printf("soskey runing!\r\n");

if(KEY0==0)

{

OSSemPost(BJSTARTSem);

safe.sos=(char)1;

}else safe.sos=(char)0;

delay\_ms(100);

OSTimeDlyHMSM(0,0,0,100);

}

}

#endif

#include "PIONEER\_Config.h"

#if (TASK\_BJ\_EN == 1)

#include "STM32F4\_Head.h"

#include "OSVars.h"

#include "UserVars.h"

#include "string.h"

void bj\_task(void \*pdata)

{

INT8U err,err1;

pdata=pdata;

while(1)

{

printf("bj runing!\r\n");

OSSemPend(BJSTARTSem,0,&err);

if(err==OS\_ERR\_NONE)

{

BEEP=1;

LED0=0;

}

OSSemPend(BJOUTSem,0,&err1);

if(err1==OS\_ERR\_NONE)

{

BEEP=0;

LED0=1;

}

OSTimeDlyHMSM(0,0,0,100);

}

}

#endif

#include "PIONEER\_Config.h"

#if (TASK\_AO\_EN == 1)

#include "STM32F4\_Head.h"

#include "OSVars.h"

#include "UserVars.h"

#include "string.h"

void ao\_task(void \*pdata)

{

threshhold safe;

u16 value\_AD = 0;

long PRESS\_AO = 0;

int VOLTAGE\_AO = 0;

pdata=pdata;

while(1)

{

printf("ao runing!\r\n");

value\_AD = Get\_Adc\_Average(1,10);

VOLTAGE\_AO = map(value\_AD, 0, 4095, 0, 3300);

if(VOLTAGE\_AO < 100)

{

PRESS\_AO = 0;

}

else if(VOLTAGE\_AO > 3300)

{

PRESS\_AO = 3300;

}

else

{

PRESS\_AO = map(VOLTAGE\_AO, VOLTAGE\_MIN, VOLTAGE\_MAX, 100, 3300);

}

printf("ADÖµ = %d,µçÑ¹ = %d mv,Ñ¹Á¦ = %ld g\r\n",value\_AD,VOLTAGE\_AO,PRESS\_AO);

if(PRESS\_AO<1)

{

OSSemPost(BJSTARTSem);

safe.pressure=(char)1;

}else

{

safe.pressure=(char)0;

}

OSTimeDlyHMSM(0,0,0,100);

}

}

#endif

#include "PIONEER\_Config.h"

#if (TASK\_MLX\_EN == 1)

#include "STM32F4\_Head.h"

#include "OSVars.h"

#include "UserVars.h"

#include "string.h"

void mlx\_task(void \*pdata)

{

threshhold safe;

int i;

int j=0;

float sum=0;

u8 t = 0;

float temp;

float sz[50];

SMBus\_Init();

pdata=pdata;

while(1)

{

printf("mlx runing!\r\n");

t++;

if(t>90)

{

t=0;

for(i=0;i<50;i++)

{

temp=SMBus\_ReadTemp();

sz[i]=temp;

OSTimeDlyHMSM(0,0,0,30);

}

for(i=0;i<50;i++)

{

if(sz[i]>30&&sz[i]<45)

{

j++;

sum+=sz[i];

}

}

temp=sum/j;

if(temp>30&&temp<45)

{

printf("%f\r\n",temp);

safe.tem=temp;

if(temp>40&&temp<34)

{

OSSemPost(BJSTARTSem);

}

}

j=0;

sum=0;

OSTimeDlyHMSM(0,0,0,50);

}

OSTimeDlyHMSM(0,0,0,100);

}

}

#endif