一、LED灯（hal_led.c）

1.函数使用

> **文件位置：**board/hal_led.c

初始化LED灯

```
void halLedInit()
```

说明：

- LED灯初始化函数

开启LED灯

```
void halLedSet(uint8 id)
```

说明：

- 开启指定LED灯

参数：

- id：1~4
  - 1：D3（P1_0）
  - 2：D4（P1_1）
  - 3：D5（P1_3）
  - 4：D6（P1_4）

关闭LED灯

```
void halLedClear(uint8 id)
```

说明：

- 关闭指定LED灯

参数：

- id：1~4（同上）

翻转LED灯

```
void halLedToggle(uint8 id)
```

说明：

- 翻转指定LED灯

参数：

- id：1~4（同上）

2.实例演示

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"  //板子头文件
#include "hal_led.h"    //LED灯定义

void main(void)
{
    halLedInit(); //LED初始化函数
    while(1)
    {
        halMcuWaitMs(1000);   //延时1秒
        halLedSet(1);         //点亮LED1
        halMcuWaitMs(1000);   //延时1秒
        halLedClear(1);       //关闭LED1
        halMcuWaitMs(1000);   //延时1秒
        halLedToggle(2);      //翻转LED2
    }
}
```

二、MCU源（hal_mcu.c）

1.函数使用

> **文件位置：**common/hal_mcu.c

初始化MCU

```
void halMcuInit(void)
```

说明：

- “主时钟源”设置为“XOSC”

微妙延时

```
void halMcuWaitUs(uint16 usec)
```

说明:

- 忙等待函数，等待指定的微秒数。使用各种指令所需的时钟周期数的假设（该函数假设时钟为32mhz）

参数:

- usec：延时的微秒数

毫秒延时

```
void halMcuWaitMs(uint16 msec)
```

说明:

- 忙等待函数，等待指定的微秒数。使用各种指令所需的时钟周期数的假设

参数:

- msec：延时的毫秒数

MCU低功耗模式设置

```
void halMcuSetLowPowerMode(uint8 mode)
```

说明:

- 设置MCU低功耗模式

参数:

- mode：模式号

MCU状态重置

```
void halMcuReset(void)
```

说明:

- 重置MCU默认状态

2.实例演示

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "hal_mcu.h"    //MCU头文件
#include "hal_led.h"

void main(void)
{
    halLedInit();
    while(1)
    {
        halMcuWaitMs(1000);   //延时1秒
        halLedSet(1);
        halMcuWaitMs(1000);   //延时1秒
        halLedClear(1);
        halMcuWaitMs(1000);   //延时1秒
        halLedToggle(2);
    }
}
```

三、SW按键

1.寄存器初始化

SW1初始化

```
P1SEL &= ~0X04;
P1DIR &= ~0X04;
```

SW2初始化

```
P0SEL &= ~0X02;
P0DIR &= ~0X02;
```

2.实例演示

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "hal_led.h"
```

```
void main(void)
{
    halLedInit();   //LED初始化

    P1SEL &= ~0X04; //SW1寄存器初始化
    P1DIR &= ~0X04;
    P0SEL &= ~0X02; //SW2寄存器初始化
    P0DIR &= ~0X02;

    while(1)
    {
        if(P1_2 == 0)   //SW1按钮触发判断
        {
            halMcuWaitMs(10);   //按键消除抖动
            if(P1_2 == 0)
            {
                while(P1_2 == 0);   //等待按键松开
                halLedToggle(1);    //按键触发内容
            }
        }
        if(P0_1 == 0)   //SW2按钮触发判断
        {
            halMcuWaitMs(10);   //按键消除抖动
            if(P0_1 == 0)
            {
                while(P0_1 == 0);   //等待按键松开
                halLedToggle(2);    //按键触发内容
            }
        }
    }
}
```

四、串口通信（hal_uart.c)

1.函数使用

> **文件位置**：common/hal_uart.c|hal_uart1.c
>
> **注**：串口1 Tx（P1.6）、Rx （P1.7）、GND

初始化串口

```
void halUartInit(uint32 baud)
```

```
void halUart1Init(uint32 baud)
```

说明：

- 初始化串口

参数:

- baud：波特率

读取Rx缓冲区数据

```
uint16 halUartRead(uint8 *buf, uint16 len)
```

```
uint16 halUart1Read(uint8 *buf, uint16 len)
```

说明:

- 读取串口Rx缓冲区数据

参数:

- buf：Rx缓冲区数据指针
- len：读取数据长度

返回:

- uint16：读取Rx缓冲区的长度

写入Tx缓冲区数据

```
uint16 halUartWrite(uint8 *buf, uint16 len)
```

```
uint16 halUart1Write(uint8 *buf, uint16 len)
```

说明:

- 写入串口Tx缓冲区数据

参数:

- buf：Tx缓冲区数据指针
- len：写入数据长度

返回:

- uint16：写入Tx缓冲区的长度

读取Rx缓冲区长度

```
uint16 halUartRxLen(void)
```

```
uint16 halUart1RxLen(void)
```

说明:

- 读取串口Rx缓冲区的有效长度

返回：

- uint16：Rx缓冲区的有效长度

2.实例演示

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"

void main(void)
{
    halBoardInit();

    while(1)
    {
        char a[128];    //定义最大接收发送最大为128
        int len = halUartRxLen();       //获取有效接收数据长度
        halUartRead((uint8 *)a,len);    //读取串口数据
        halUartWrite((uint8 *)a,len);  //发送串口数据
        //串口1同串口0只需要更改函数名称
    }
}
```

五、定时器（hal_timer_32k.c）

1.函数使用

> **文件位置：**common/hal_timer_32k.c

初始化定时器

```
void halTimer32kInit(uint16 cycles)
```

说明：

- 初始化定时器

参数：

- cycles：周期数，3125对应0.1s（1~8191）

重置定时器

```
void halTimer32kRestart(void)
```

说明：

- 重置定时器计数为0，并重启定时器

函数中断设置

```
void halTimer32kIntConnect(ISR_FUNC_PTR isr)
```

说明：

- 将函数连连接到定时器中断

参数：

- isr：函数

开启定时器中断

```
void halTimer32kIntEnable(void)
```

说明：

- 使能32KHz定时器中断

关闭定时器中断

```
void halTimer32kIntDisable(void)
```

说明：

- 禁用32KHz定时器中断

2.实例演示

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "hal_led.h"
#include "hal_timer_32k.h"  //定时器头文件

int a = 0;
```

```
void Timer(void)
{
    a++;
    if(a == 10)
    {
        halLedToggle(1);
        a = 0;
    }
}

void main(void)
{
    halBoardInit();
    halLedInit();
    halTimer32kInit(3125);           //定时器初始化
    halTimer32kIntConnect(Timer);    //绑定中断函数
    halTimer32kIntEnable();          //启动定时器中断
}
```

六、传感器（sensor/）

1.函数使用

> **文件位置**：sensor/get_swsensor.c

开关量传感器

读取开关量值（P0_4）

```
uint8 get_swsensor(void)
```

说明：

- 读取开关量值

返回：

- uint8：开关量值

> **注**：这函数简直是脱了裤子放屁——多此一举

开关量执行器

> **文件位置**：sensor/get_adc.c

模拟量传感器（ADC）

初始化ADC

```
void hal_adc_Init(void)
```

说明：

- 初始化ADC端口

读取ADC值（P0_0）

```
float get_adc(void)
```

说明：

- 读取ADC端口电压值

返回：

- float：电压值

初始化四输入

```
void hal_adc4CH_Init(void)
```

说明：

- 初始化ADC四输入

读取四输入值

```
float get_adc4CH(uint8 ch)
```

说明：

- 读取ADC四输入值

参数：

- ch：四输入端口
  - 0：IN1
  - 1：IN2
  - 2：IN3
  - 3：IN4

返回：

- float：对应端口电压值

> **文件位置**：sensor/sht.c

模拟量传感器（I2C）

初始化温湿度

```
void SHT_Init(void)
```

说明：

- 初始化温湿度传感器

读取温湿度值

```
void SHT_SmpSnValue(float *tem, float *hum)
```

```
void call_sht11(float *tem, float *hum)
```

说明：

- 读取温湿度传感器值

输出：

- tem：温度
- hum：湿度

2.实例演示

开关量传感器

> **适用传感器**：人体传感器

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "get_swsensor.h"   //开关量传感器头文件

void main(void)
{
    int Value = get_swsensor(); //人体红外值（0~1）
}
```

模拟量传感器（ADC）

> **计算公式**：传感器实际值=(实际电压-最小电压)*(传感器最大值-传感器最小值)/(最大电压-最小电压)+传感器最小值

> **适用传感器**：光照传感器、空气质量传感器、可燃气传感器、火焰传感器
>
> **最小电压**：0V　**最大电压**：3.3V

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "get_adc.h"    //ADC头文件

void main(void)
{
    halBoardInit(); //板子初始化
    hal_adc_Init(); //ADC初始化

    float ValueLight = get_adc()*850/3.3;        //光照（0~850）lx
    float ValueAir = get_adc()*850/3.3;          //空气质量(0~850)ppm
    float ValueFlammable = get_adc()*850/3.3;    //可燃气(0~100)ppm
    float ValueFire = get_adc()*2/3.3;           //火焰(0~1)
}
```

> **适用传感器**：四输入模拟量、两输入模拟量?
>
> **最小电压**：0.58V~0.66V　**最大电压**：2.9V~3.3V

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "get_adc.h"    //ADC头文件

void main(void)
{
    hal_adc4CH_Init(); //四输入初始化函数

    char ValueStr[5];
    float IN1 = get_adc4CH(0);  //读取IN1
    float ValueLight = (IN1 -0.58)*20000/(2.9-0.58); //光照换算
    float ValueLight = (IN1 -0.66)*20000/(3.3-0.66); //光照换算
}
```

模拟量传感器（I2C）

> **适用传感器**：温湿度传感器

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_board.h"
#include "sht.h"    //温湿度头文件

void main(void)
{
    SHT_Init(); //温湿度初始化函数

    float ValueTemp, ValueHum;
    SHT_SmpSnValue(&ValueTemp, &ValueHum);  //货期温湿度值
}
```

七、点对点（basic_rf.c)

1.函数使用

> **文件位置：**basicrf/basic_rf.c

初始化BasicRF

```
unsigned char basicRfInit(basicRfCfg_t *pRfConfig);
```

说明：

- 初始化点对点BasicRF

参数：

- pRfConfig：BasicRF结构体

返回：

- unsigned char：是否成功

发送数据

```
unsigned char basicRfSendPacket( unsigned short destAddr, unsigned char
*pPayload, unsigned char length);
```

说明：

- 向指定节点发送数据

参数：

- destAddr：目标地址
- pPayload：数据地址
- length：发送长度

返回:

- unsigned char：是否成功

接收检测

```
unsigned char basicRfPacketIsReady(void);
```

说明:

- 检测是否有数据接收

返回:

- 是否有数据

接收数据

```
unsigned char basicRfReceive(unsigned char *pRxData,  unsigned short len,  short *pRssi);
```

说明:

- 接收发送过来的数据

参数:

- pRxData：数据地址
- len：接收长度
- pRssi：型号强度（NULL）

开启接收

```
void basicRfReceiveOn(void);
```

说明:

- 开启接收功能

关闭接收

```
void basicRfReceiveOff(void);
```

说明:

- 关闭接收功能

> **文件位置：**basicrf/lib_radio.c

初始化点对点

```
void Radio_Init(uint16 panId,uint8 channel,uint16 addr)
```

说明：

- 初始化点对点（其实里面套了一个`basicRfInit`）

参数：

- panId：频道号（11~26）
- channel：网络号（0x0000~0xFFFF）
- addr：本机地址（0x0000~0xFFFF）

2.实例演示

```
好像有问题不能用
```

八、呼吸灯（hal_pwm.c）

1.函数使用

> **文件位置：**board/hal_pwm.c

初始化PWM

```
void TIM1_PwmInit(uint16 period, uint8 ration)
```

说明：

- 设置PWM时长及占空比

参数：

- period：周期时长
- ration：占空比

> 但貌似只能控制LED1

2.实例演示

```
#include "hal_defs.h"
#include "hal_cc8051.h"
#include "hal_pwm.h"
#include "hal_mcu.h"

int main(void)
{
    while(1)
    {
        for(int i = 100; i > 0; i--)
        {
            TIM1_PwmInit(1, i);
            halMcuWaitMs(10);
        }
        for(int i = 1; i < 100; i++)
        {
            TIM1_PwmInit(1, i);
            halMcuWaitMs(10);
        }
    }
}
```

九、Flash存储

1.函数使用

不会

2.实例演示

不会