外接模块驱动说明例程

——望穿秋水

一、使用说明

1.在 main()上方加入头文件：#include “ds18b20.h”,并在工程左侧栏的 Driver 文件夹加入 ds18b20.c、ds18b20.h 共二个文件，接口在初使化函数设置,具体使用查看 ds18b20.h。

2.IIC 硬件接口在 i2c_gpio.h 里设置。(IIC 无需初使化)

二、例程

<1>DS18B20-温度传感器

例一：

(1) 在 ds18b20.h 中修改数量，这里假设为 2

```
#define DS18B20_NUM        2                  //DS18B20 数量
```

(2) 在 main()中初使化

```
void main(void)
{
   /*-----------------全局初使化------------------*/
       DS18B20_Config(GPIOB,GPIO_Pin_12);   //对应温度结构体数据 0 组
       DS18B20_Config(GPIOB,GPIO_Pin_6);    //对应温度结构体数据 1 组
    /*-------------------------------------------------------*/
}
void Task1(void)   //任务 1
{
   while(1)
    {
        OSSchedLock();   //在读取传感器时禁止任务切换
        DS18B20_Update(GPIOB,GPIO_Pin_12);
        DS18B20_Update(GPIOB,GPIO_Pin_6);
        OSprintf("PB14_温度: %0.2f \r\n",Ds18b20.Val [0]);
        OSprintf("PB8 _温度: %0.2f \r\n",Ds18b20.Val[1]);
        OSprintf("PB14 _最高温度: %0.2f \r\n",Ds18b20.ValMax[0]);
       OSprintf("PB14 _最低温度: %0.2f \r\n",Ds18b20.ValMin[0]);
       OSprintf("PB8 _最高温度: %0.2f \r\n",Ds18b20.ValMax[1]);
       OSprintf("PB8 _最低温度: %0.2f \r\n",Ds18b20.ValMin[1]);
      OSSchedUnlock(); //读取完闭打开任务切换
      OS_delayMs(1000); //示例代码，使用时删除
    }
}
```

例二：

```
例：#define DS18B20_NUM        2        //设为 2 个

void Task1(void)      //任务 1 用作更新数据
{
```

```
    DS18B20_Config(GPIOB,GPIO_Pin_12); //第一个初使化，PB12 对应结构体数组 0
    DS18B20_Config(GPIOB,GPIO_Pin_6);  //第二个初使化，PB6 对应结构体数组 1
    while(1)
    {
        OSSchedLock();//读取数据时，一定要关闭任务切换，防止切换到下个任务
        DS18B20_Update(GPIOB,GPIO_Pin_12); //更新温度数据，更新数据时 IO 口无顺序要
求
        DS18B20_Update(GPIOB,GPIO_Pin_6);
        OSSchedUnlock();//读取数据完闭，恢复任务切换
        OS_delayMs(100); //100ms 更新一次温度
    }
}
void Task2(void)   //任务 2 用作处理数据
{
  while(1)
  {
    OSprintf("PB12 当前温度：%0.1f℃   \r\n",Ds18b20.Val[0]);//数组 0 对应 PB12 的温度
    OSprintf("PB6 当前温度：  %0.1f℃   \r\n",Ds18b20.Val[1]); //数组 1 对应 PB6 的温度

    if(Ds18b20.Val[1]>30.0) //判断第二个初使化的 DS118B20 温度，即 PB6 线上的 DS18B20
    {
      OSprintf("PB6 温度超高！    \r\n");
      //xxxxx();//关闭加热器(xxxx()为关闭加热函数，根据实际加热器自己写)
    }
    OS_delayMs(1000);//1 秒查询一次
  }
}
```

<2>DS3231-高精度实时时钟 IC (IIC 通讯)

例一：

在 main()上方加入头文件：#include “ds3231.h”,并在工程左侧栏的 Driver 文件夹加入 ds3231.c、ds3231.h、i2c_gpio.c、i2c_gpio.h 共四个文件

```
void Task1(void)   //任务 1
{
  OSSchedLock();//锁定任务切换
  DS3231_Config();//初使化 DS3231（主要用于检测 DS3231 是否正常）
  DS3231_SetTime(2019,01,20,7,19,26,30); //设置初使时间(如 IC 时间是准的，可不用设置)
  OSSchedUnlock();//解锁任务切换
  while(1)
     {
       OSSchedLock();  //在读数据时上锁任务切换
       DS3231_Update(&CurrentTime);   //更新时间
       if(CurrentTime.SecCount!=CurrentTime.Second)
```

```
        {
          CurrentTime.SecCount=CurrentTime.Second;
          OSprintf("%d 年%d 月%d 日
          \r\n",CurrentTime.Year+2000,CurrentTime.Month,CurrentTime.Day);
          OSprintf("%d 时%d 分%d 秒
          \r\n",CurrentTime.Hour,CurrentTime.Minute,CurrentTime.Second);
          OSprintf("星期%d\r\n\r\n",CurrentTime.Week);
        }
          OSSchedUnlock();//在读取完数据解锁任务切换
          OS_delayMs(1000);//延时 1s
      }
}
```

<3>AT24Cxx EEPROM 断电数据保存 IC　(IIC 通讯)

准备工作：
容量设置：在 eeprom_24xx.h 里设置容量大小，并修改对应容量的地址为器件实际地址。24C02-24C16 选择#define AT24C02，大于 24C16 选择#define AT24C128，大于 128 请自行修改 128 对应的参数。

例一：单个数据或数组的保存

```
uint8_t temp[10];   //全局变量

void Task2(void) //任务 2
{
   uint8_t arr[10];
   OSSchedLock();//锁定任务切换
   EEPROM_Config(); //初使化 eeprom
   temp[0]=32;
   temp[1]=35;
   temp[2]=39;
   temp[3]=87;
   EEPROM_WriteBytes(temp, 0x0001, 4);//从 eeprom 的 0001 地址(也可以为其他地址)开始连
//续写入 4 个数据
   OSSchedUnlock();//解锁任务切换
   while(1)
   {
      OSSchedLock();//锁定任务切换
      EEPROM_ReadBytes(arr, 0x0001, 4);//从 eeprom 的 0001 地址开始连续读取 4 个数据保存
      //到 arr[]数组
      OSprintf("arr[0]-arr[3]= %d %d %d %d\r\n",arr[0],arr[1],arr[2],arr[3]);//显示 arr 数组的 4 个
      //成员的值
      OSSchedUnlock();//解锁任务切换
```

```
    OS_delayMs(1000);//1 秒读取一次
  }
}
```

写入数据后，把上面的写数据注释掉，让 eeprom 断电一次，然后从 eeprom 读取数据：

```
void Task2(void) //任务 2
{
  uint8_t arr[10];
  OSSchedLock();//锁定任务切换
  EEPROM_Config();
//    temp[0]=32;   //注释
//    temp[1]=35;   //注释
//    temp[2]=39;   //注释
//    temp[3]=87;   //注释
//    EEPROM_WriteBytes(temp, 0x0001, 4);//从 eeprom 的 0001 地址(也可以为其他地址)开始
//连续写入 4 个数据   //注释
  OSSchedUnlock();//解锁任务切换
  while(1)
  {
    OSSchedLock();//锁定任务切换
    EEPROM_ReadBytes(arr, 0x0001, 4);//从 eeprom 的 0001 地址开始连续读取 4 个数据保存
//到 arr[]数组
    OSprintf("arr[0]-arr[3]= %d %d %d %d\r\n",arr[0],arr[1],arr[2],arr[3]);//显示 arr 数组的 4 个
//成员的值
    OSSchedUnlock();//解锁任务切换
    OS_delayMs(1000);//1 秒读取一次
  }
}
```

发现读取出的值仍未改变，说明写入成功。

例二：结构体数据的保存(推荐使用结构体方式创建需要保存的重要数据)

```
typedef struct
{
  uint8_t tee;
    uint32_t neot;
    uint32_t weight;
}TypeDefSaveData;
TypeDefSaveData NeedSaveData;
```

```
void Task2(void) //任务 2
{
    TypeDefSaveData arr;//缓存读取的值用的临时变量
    OSSchedLock();//锁定任务切换
  EEPROM_Config();
    NeedSaveData.tee=56;
    NeedSaveData.neot=3564;
    NeedSaveData.weight=835447;
    EEPROM_WriteBytes((uint8_t*)(&NeedSaveData), 0x0001, sizeof(NeedSaveData));
    OSSchedUnlock();//解锁任务切换
    while(1)
     {
          OSSchedLock();//锁定任务切换
          EEPROM_ReadBytes((uint8_t*)(&arr), 0x0001, sizeof(NeedSaveData));
          OSprintf("arr[0]-arr[2]= %d %d %d \r\n",arr.tee,arr.neot,arr.weight);
          OSSchedUnlock();//解锁任务切换
        OS_delayMs(1000);//1 秒读取一次
     }
}
```

上面代码是测试，如果使用于实际应用中，开机时需要从 eeprom 中读取出数据并保存到原来的变量中，即等于恢复数据

```
typedef struct
{
  uint8_t tee;
  uint32_t neot;
  uint32_t weight;
}TypeDefSaveData;
TypeDefSaveData NeedSaveData;

void Task2(void) //任务 2
{
    OSSchedLock();//锁定任务切换
    EEPROM_Config();
    NeedSaveData.tee=56;
    NeedSaveData.neot=3564;
    NeedSaveData.weight=835447;
    EEPROM_WriteBytes((uint8_t*)(&NeedSaveData), 0x0001, sizeof(NeedSaveData));
    OSSchedUnlock();//解锁任务切换
    while(1)
     {
          OSSchedLock();//锁定任务切换
```

```
            EEPROM_ReadBytes((uint8_t*)(&NeedSaveData), 0x0001, sizeof(NeedSaveData));
            OSprintf("数据  tee    neot    weight = %d %d %d
            \r\n",NeedSaveData.tee,NeedSaveData.neot,NeedSaveData.weight);
            OSSchedUnlock();//解锁任务切换
            OS_delayMs(1000);//1 秒读取一次
        }
}
```

读写函数返回值可以表示读写是否成功(返回 0-失败　1-成功)，可以改为:

```
typedef struct
{
   uint8_t tee;
   uint32_t neot;
   uint32_t weight;
}TypeDefSaveData;
TypeDefSaveData NeedSaveData;

void Task2(void) //任务 2
{
   OSSchedLock();//锁定任务切换
   EEPROM_Config();
//    NeedSaveData.tee=56;
//    NeedSaveData.neot=3564;
//    NeedSaveData.weight=835447;
//    EEPROM_WriteBytes((uint8_t*)(&NeedSaveData), 0x0001, sizeof(NeedSaveData));

    if(EEPROM_ReadBytes((uint8_t*)(&NeedSaveData), 0x0001, sizeof(NeedSaveData))==1) //只
    //需要在开机时恢复一次
    {
          OSprintf("数据恢复成功！  \r\n");
    }
    else
    {
        OSprintf("数据恢复失败！  \r\n");
    }
    OSprintf("数据  tee    neot    weight = %d %d %d
    \r\n",NeedSaveData.tee,NeedSaveData.neot,NeedSaveData.weight);
    OSSchedUnlock();//解锁任务切换
    while(1)
     {
        OSSchedLock();//锁定任务切换
        EEPROM_WriteBytes((uint8_t*)(&NeedSaveData), 0x0001, sizeof(NeedSaveData));//定
        //时保存或判断数据变化了再保存
```

```
            OSSchedUnlock();//解锁任务切换
            OS_delayMs(1000);//1 秒保存一次
        }
}
```