# 1.E53\_SC1扩展板及其驱动

关于E53标准接口

E53接口标准的E取自扩展（Expansion）的英文首字母，板子的尺寸为5×3cm，故采用E53作为前缀来命名尺寸为 5×3cm 类型的案例扩展板，任何一款满足标准设计的开发板均可直接适配E53扩展板。

E53扩展板是根据不同的应用场景来设计的，以最大的程度在扩展板上还原真实应用场景，不同案例的扩展板根据不同的应用场景来命名后缀。例如：E53\_SC1，SC是智慧城市（Smart City）的缩写，SC1表示的是智慧城市中的智慧路灯，再比如SC2则表示的是智慧城市中的智慧井盖。

E53扩展接口在电气特性上，包含了常用的物联网感知层传感器通信接口，比如5V、3.3V、GND、SPI、UART、IIC、ADC、DAC等等，可以适配各种传感器，还留有4个普通GPIO，如图：

E53接口电气特性

E53\_SC1智慧路灯扩展板

E53\_SC1扩展板采用了E53标准接口，包含了一个路灯灯珠，一个BH1750光照强度传感器，其中路灯灯珠使用普通GPIO控制，BH1750使用IIC接口通信。

E53\_SC1扩展板

# 2. 移植E53\_SC1驱动到LiteOS

复制裸机驱动文件到LiteOS工程

E53\_SC1扩展板上的 BH1750 光照强度传感器使用的是 IIC 通信接口，所以除了复制 STM32CubeMX 生成的i2c.h和i2c.h文件，还需要在此基础上复制包含了 BH1750 传感器驱动的 E53\_SC1 扩展板驱动文件。

在复制文件的时候，按照上一篇文章中所说的，复制i2c.h到Inc 文件夹，复制i2c.c到 Src 文件夹，再复制自己编写的驱动文件E53\_SC1.c、E53\_SC1.h到 Hardware文件夹。

IoT-Studio中提供的默认工程已经复制好了这些文件，无需再次添加，如图：

E53\_SC1扩展板驱动

添加驱动文件路径

因为 LiteOS 的整个项目工程使用 make 构建，所以复制驱动文件之后，需要添加驱动文件的路径到 makefile 中，加入编译。

project.mk文件指明了工程中所有文件的路径：

project.mk文件

在该文件中：

C文件路径

HARDWARE\_SRC：对应Hardware文件夹下的Src文件夹

USER\_SRC：对应Src文件夹

头文件路径

HARDWARE\_INC：对应Hardware文件夹下的Inc文件夹

USER\_INC：对应Inc文件夹

如下，E53\_SC1 驱动的底层 I2C 接口代码i2c.c路径添加到USER\_SRC中：

I2C驱动代码

E53\_SC1 驱动的底层 I2C 接口代码i2c.h路径添加到USER\_INC中：

I2C驱动头文件路径

基于 I2C 驱动的 E53\_SC1 驱动文件E53\_SC1.c添加到HARDWARE\_SRC中(默认未添加，需要手动添加)：

E53\_SC1驱动文件

基于 I2C 驱动的 E53\_SC1 驱动文件E53\_SC1.h添加到 HARDWARE\_INC 中(默认未添加，需要手动添加)：

E53\_SC1驱动头文件路径

至此，复制文件到LiteOS工程中，并将新复制的文件路径添加到makefile中，加入工程编译，就完成了驱动的移植。

# 3. E53\_SC1裸机驱动的使用

初始化E53\_SC1扩展板

在上一篇文章中详细的讲述了在LiteOS中初始化设备的两种方式：

在系统启动调度之前初始化：设备在系统中随时可被任意任务使用

在任务中初始化：设备一般只在该任务中被使用

本文中移植的 E53\_SC1 扩展板驱动，不需要多个任务去操作，只需要传感器数据采集任务操作即可，所以初始化放在数据采集任务中。

操作E53\_SC1扩展板

接下来首先创建一个文件夹（如果已有，不用再次创建），用于存放本系列教程实验的代码：

新建文件夹

文件夹名称

在该文件夹中创建一个文件：

创建Demo文件

编写代码:

#include <osal.h>

#include "lcd.h"

#include "E53\_SC1.h"

/\* 存放E53\_SC1扩展板传感器数据，可在E53\_SC1.h中查看定义 \*/

E53\_SC1\_Data\_TypeDef E53\_SC1\_Data;

/\* 用于数据采集和数据处理任务间同步的信号量 \*/

osal\_semp\_t sync\_semp;

/\* 数据采集任务-低优先级 \*/

static int data\_collect\_task\_entry()

{

/\* 初始化扩展板 \*/

Init\_E53\_SC1();

while (1)

{

/\* 读取扩展板板载数据，存到数据结构体E53\_SC1\_Data中 \*/

E53\_SC1\_Read\_Data();

/\* 数据读取完毕，释放信号量，唤醒数据处理任务 \*/

osal\_semp\_post(sync\_semp);

/\* 任务睡眠2s \*/

osal\_task\_sleep(2\*1000);

}

}

/\* 数据处理任务-高优先级 \*/

static int data\_deal\_task\_entry()

{

/\* lux- 当次数据，old-lux-上次数据 \*/

int lux = 0, old\_lux = 0;

/\* LCD清屏，防止干扰显示 \*/

LCD\_Clear(WHITE);

while (1)

{

/\* 等待信号量，未等到说明数据还未采集，阻塞等待 \*/

osal\_semp\_pend(sync\_semp, cn\_osal\_timeout\_forever);

/\* 信号量等待，被唤醒，开始处理数据 \*/

old\_lux = lux;

lux = (int)E53\_SC1\_Data.Lux;

printf("BH1750 Value is %d\r\n", lux);

LCD\_ShowString(10, 100, 200, 16, 16, "BH1750 Value is:");

LCD\_ShowNum(140, 100, lux, 5, 16);

/\* 阈值为1000，自动点亮或者熄灭路灯 \*/

if(old\_lux < 1000 && lux > 1000)

{

HAL\_GPIO\_WritePin(SC1\_Light\_GPIO\_Port, SC1\_Light\_Pin, GPIO\_PIN\_RESET);

printf("Street Light OFF!\r\n");

}

else if(old\_lux > 1000 && lux < 1000)

{

HAL\_GPIO\_WritePin(SC1\_Light\_GPIO\_Port, SC1\_Light\_Pin, GPIO\_PIN\_SET);

printf("Street Light ON!\r\n");

}

}

}

/\* 标准demo启动函数，函数名不要修改，否则会影响下一步实验 \*/

int standard\_app\_demo\_main()

{

/\* 创建信号量 \*/

osal\_semp\_create(&sync\_semp, 1, 0);

/\* 数据处理任务的优先级应高于数据采集任务 \*/

osal\_task\_create("data\_collect",data\_collect\_task\_entry,NULL,0x400,NULL,3);

osal\_task\_create("data\_deal",data\_deal\_task\_entry,NULL,0x400,NULL,2);

return 0;

}

然后按照之前的方法，在 user\_demo.mk 中将lcd\_driver\_demo.c文件添加到makefile中，加入编译:

user\_demo.mk文件

最后在.sdkconfig中配置开启宏定义：

.sdkconfig文件

编译，烧录，即可看到实验现象。

LCD屏幕上显示当前传感器采集的亮度值，并且每2s更新一次，当亮度值低于1000时，E53\_SC1扩展板的路灯自动点亮：

路灯自动点亮

当亮度值高于1000时，E53\_SC1扩展板的路灯自动熄灭：

路灯自动熄灭

另外，打开IoT-Studio自带的串口终端，可以查看到串口输出的工作信息：

linkmain:V1.2.1 AT 10:40:09 ON Dec 5 2019

BH1750 Value is 220

WELCOME TO IOT\_LINK SHELL

LiteOS:/>BH1750 Value is 612

BH1750 Value is 566

BH1750 Value is 14109

Street Light OFF!

BH1750 Value is 14814

BH1750 Value is 178

Street Light ON!

BH1750 Value is 179

……