BabyOS 设计和使用手册

源码地址

https://github.com/notrynohigh/BabyOS

https://gitee.com/notrynohigh/BabyOS

开发者 QQ 群

管理员邮箱

notrynohigh@outlook.com

修订记录

时间	记录	修订人
2020.02.19	1.创建文档	notrynohigh
2020.02.24	1.更新功能模块描述	notrynohigh
2020.03.05	1.新增 gui，log,menu 的描述	notrynohigh
2020.03.11	1.改变文档结构	notrynohigh

目录

1. 引言

BabyOS 是为 MCU 裸机项目而生，分为驱动和功能模块两个部分。本文档介绍 BabyOS 的设计以及使用方法，作为开发者优化代码框架和新增代码的参考。

希望在日后的某一天它能成为工程师喜爱的一份代码。

2. 开发组成员

Notrynohigh

不愿透漏姓名的王年年

超级布灵的小星星

Cloud

段仁胜

Illusion

绿色心晴

Lyping

Murphy

嵌入式_蓝莲花

思无邪

无诚无成

蜗牛

向日葵

.

3. 设计思路

BabyOS 的定位在 MCU 裸机开发，而复杂的项目可以选择经过大量验证的操作系统 FreeRTOS ,uCOS-II/III 等。

一个公司开发的产品之间会有较多相同的功能，例如智能穿戴设备公司，以手环和智能跑鞋为例，除显示和算法不同，其他功能几乎都是可以复用的。可复用的功能以功能模块的形式存在于 BabyOS,新项目开始时可通过搭积木的方式选择已有的功能模块。以这种方式减少重复的工作加快项目开发。

物联网领域使用 MCU 进行裸机开发的产品非常多，物联网其中一个非常重要的特性是低功耗。为方便工程师控制功耗，BabyOS 的驱动操作设计为类似文件的操作，以打开和关闭文件对应设备的唤醒和睡眠。从功能模块的角度考虑，驱动使用了统一的接口也方便了功能模块的设计。

3.1. 代码结构

图 3-1 BabyOS 代码目录

Core:核心以及功能模块

Drivers:驱动代码

Hal:硬件相关代码

移植过程：

1) Core 目录代码全部添加至工程，通过 b_config.h 文件配置功能模块
2) 选择 Drivers 目录下需要的驱动代码添加至工程
3) b_device_list.h 内注册设备
4) 将硬件相关的代码编写在 Hal 目录的 b_hal.c b_hal.h 中
5) 配置_TICK_FRQ_HZ 值，并根据对应频率调用 bHalIncSysTick

3.2. 设备和驱动

3.2.1. 驱动接口

BabyOS 驱动有统一的接口如下：

```
typedef struct
{
    int (*init)(void);
    int (*open)(void);
    int (*close)(void);
    int (*ctl)(uint8_t cmd, void *param);
    int (*write)(uint32_t addr, uint8_t *pbuf, uint16_t len);
    int (*read)(uint32_t addr, uint8_t *pbuf, uint16_t len);
}bDriverInterface_t;
```

1) 初始化 init，执行初始化设备的操作，成功返回 0，异常返回-1。如果初始化异常，设备会被标记为异常设备，其他操作都无法执行。

2) 打开/关闭 open/close，打开和关闭用于做设备唤醒和休眠的相关操作。

3) 控制 ctl，执行控制用于配置设备至特定模式等一些特定操作。执行函数需要带上参数 cmd 和 param，当前驱动支持的 cmd 以及 cmd 对应 param 的类型需要在 b_device.h 中定义。

4) 读/写 read/write，读写操作用于和设备进行数据交互。

3.2.2. 设备接口

初始化阶段执行 *bInit*，*bInit* 会依次执行在 b_device_list.h 注册的所有设备驱动的 init。执行失败（返回值-1）则标记设备为异常设备。

执行 *bOpen bClose bCtl bWrite bRead* 操作，最终执行其驱动的 open close ctl write read。其中 *bCtl* 需要的参数 cmd 和 param 类型在 b_device.h 内定义。

执行 *bOpen* 后会得到一个句柄（≥0），默认限制是同时打开的设备为 10 个，在 b_core.h 中修改 BCORE_FD_MAX 值进行调整。执行 *bClose* 后句柄会被回收。当某个设备处于被打开的状态则不能再次被打开，执行 *bOpen* 后必须判断句柄是否有效。

每个句柄都对应一个 bCoreFd_t 类型的结构体，其中 uint32_t lseek 成员会在执行读写操作时自加，调用驱动的读写函数时这个值会当做 addr 参数传入，改变这个值是通过 *bLseek* 完成。

```
int bOpen(uint8_t dev_no, uint8_t flag);
int bRead(int fd, uint8_t *pdata, uint16_t len);
int bWrite(int fd, uint8_t *pdata, uint16_t len);
int bCtl(int fd, uint8_t cmd, void *param);
int bLseek(int fd, uint32_t off);
int bClose(int fd);
```

3.2.3. 设备注册

在文件 b_device_list.h 内通过如下宏进行设备注册：

```
B_DEVICE_REG(W25QXX, bW25X_Driver, "flash")
```

设备号：W25QXX

驱动：bW25X_Driver

描述：flash

b_device.h 中通过如下方式统计设备数量：

```
typedef enum
{
    #define B_DEVICE_REG(dev, driver, desc)    dev,
    #include "b_device_list.h"
    bDEV_MAX_NUM
}bDeviceName_t;
```

b_device.c 中通过如下方式建立驱动数组及描述数组：

```
static bDriverInterface_t* bDriverTable[bDEV_MAX_NUM] = {
    #define B_DEVICE_REG(dev, driver, desc)    &driver,
    #include "b_device_list.h"
};

static const char *bDeviceDescTable[bDEV_MAX_NUM] = {
    #define B_DEVICE_REG(dev, driver, desc)    desc,
    #include "b_device_list.h"
};
```

设备号就是各个驱动和描述的索引，由此实现设备号和驱动的对应。

3.3. 功能模块设计

每个功能模块只做成一个功能，可通过配置文件对其进行 ENABLE/DISABLE，增加一项功能模块需要在 b_config.h 中增加一项开关。大致会有如下几种特性的功能模块：

1）用户主动调用功能模块提供的 API

2）用户提供回调函数，由功能模块调用回调

当功能模块需要使用硬件资源时，提供 API 给用户，让其指定设备号。用户指定的设备号是 3.2.3 章节提到的在 b_device_list.h 中注册的设备号。由于操作设备的接口是统一的，那么知道设备号后，功能模块便知道怎么操作设备了。

当功能模块有需要循环执行的操作时，例如检测超时等，将这些操作放入 *bExec()*函数内执行。

使用功能模块是先将 b_config.h 内对应的宏打开。

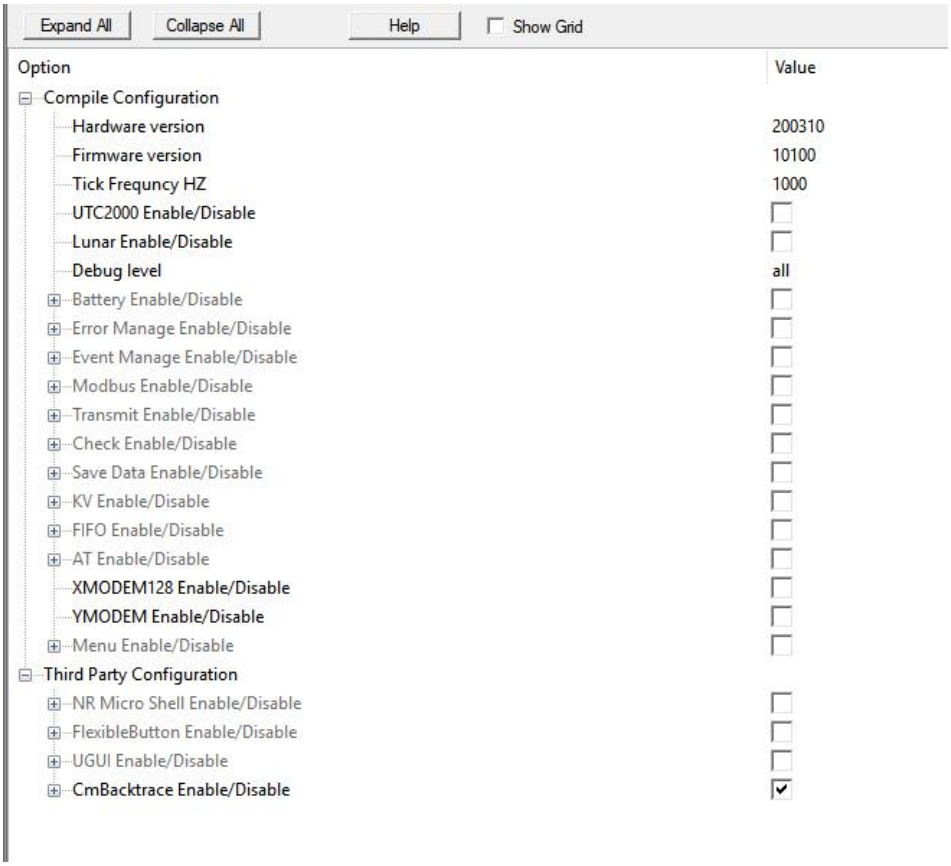

图 4-1 配置页面

3.3.1. 电量检测

3.3.2. 校验 CRC/累加和/异或

3.3.3. 错误管理

3.3.4. 事件管理

3.3.5. MODBUS RTU

3.3.6. 发送管理

3.3.7. 私有协议

3.3.8. 数据存储

3.3.9. UTC 转换

3.3.10. FIFO

3.3.11. AT

3.3.12. 阳历阴历

3.3.13. KV 键值对存储

3.3.14. Xmodem128 和 Ymodem

3.3.15. Shell

3.3.16. FlexibleButton

3.3.17. 打印日志 b_log

3.3.18. uGUI

3.3.19. 菜单程序

4. 使用教程

4.1. 详细教程

代码仓库:https://gitee.com/notrynohigh/BabyOS_Example

不同分支对应不同教程，根据需要选择不同的分支查看。

4.2. 概要描述

https://gitee.com/notrynohigh/BabyOS/wikis

https://github.com/notrynohigh/BabyOS/wiki

5. 期望

一份代码的成长离不开网络的大环境，希望能够在众多网友的支持下，将她不断的扩充不断的完善。让她成为 MCU 裸机开发中不可缺少的一部分。也希望各位开发者一起努力优化她。