

**BARE SYSTEM**

V1.0



2020-9-1

惊奇电子

# 目录

[目录 1](#_Toc17248)

[1．概要介绍 2](#_Toc9345)

[1. 1 BARE SYSTEM是什么？ 2](#_Toc29939)

[1.2和CCB的关系？ 3](#_Toc5387)

[2. 代码架构 4](#_Toc31149)

[2.1 文件组织 4](#_Toc20856)

[2.2 程序框图 4](#_Toc31668)

[2.3 框架分层介绍 4](#_Toc24599)

[2.3.1 核心层 4](#_Toc25045)

[2.3.2 片内\外驱动层 4](#_Toc1877)

[2.3.3应用层 4](#_Toc12373)

[2.3.4 bsp层 4](#_Toc23494)

[3. 实现原理 5](#_Toc19796)

[3.1 设计思路 5](#_Toc10105)

# 1．概要介绍

## 1 BARE SYSTEM是什么？

BARE SYSTEM适用于MCU项目，它是一套管理功能模块和外设驱动的框架。

**对项目而言，极大的缩短开发周期。**项目开发时选择所需要的**MCU型号**和**功能模块**后，可直接进入功能代码编写的阶段。

**对工程师而言，减少重复工作，开发过的代码不用修改重复可用。**调试过的功能模块和驱动代码以BARE SYSTEM接管，以后项目可以直接使用，去掉重复调试的工作。**且代码高度解耦，若是分工合作可互相调试自己的代码**。

**对公司产品而言，代码逐渐经过市场验证将越来越稳定，后期经过迭代，新出的产品将有扎实的基础，亦将是公司宝贵的资产。**且在BARE SYSTEM良好的分层架构下，工程师一旦熟悉后维护、修改别人的代码亦非常方便。

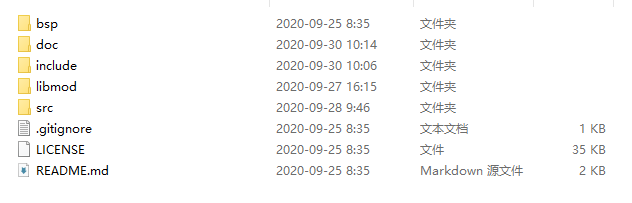
## 1.2和CCB的关系？

CCB的意思是代码模块化，任何公司的软件产品均需要实现CCB，用以增加公司产品的稳定性、可维护性，也能大大的缩减新项目的开发周期。

**BARE SYSTEM可以说是实现CCB的手段**，为实现软件仓库，BARE SYSTEM将嵌入式或单片机项目分为5层仓库，分别为**BARE SYSTEM层**、**片内\外驱动层**、**应用层**、**bsp层**，各层仓库互相更新维护自己代码，互不相干，然后通过BARE SYSTEM作为桥梁以搭积木的方式即可快速完成新项目。

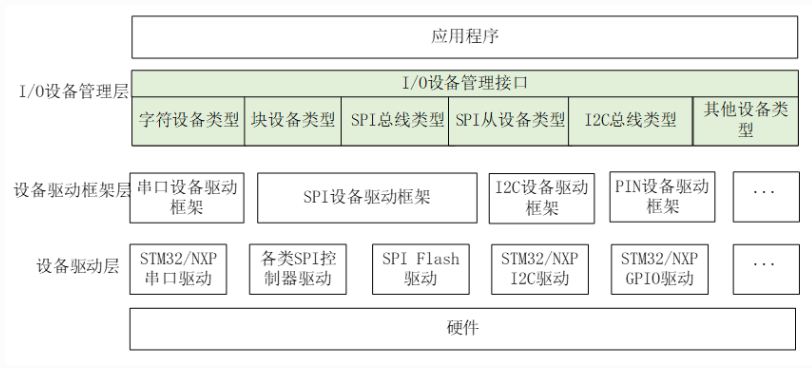
# 代码架构

## 文件组织



|  |  |
| --- | --- |
| bsp | Board support package，bare-system 板级支持包（IAR/MDK 工程在 BSP 目录下的具体的 BSP 中） |
| doc | 一些说明文件，如代码风格说明、程序架构说明 |
| include | 内核与模块的所有头文件 |
| libmod | 包含已经投入使用的片内片外驱动模块框架 |
| src | 内核代码 |

## 程序框图



用程序通过 I/O 设备管理接口获得正确的设备驱动，然后通过这个设备驱动与底层 I/O 硬件设备进行数据（或控制）交互。

I/O 设备管理层实现了对设备驱动程序的封装。应用程序通过 I/O 设备层提供的标准接口访问底层设备，设备驱动程序的升级、更替不会对上层应用产生影响。这种方式使得设备的硬件操作相关的代码能够独立于应用程序而存在，双方只需关注各自的功能实现，从而降低了代码的耦合性、复杂性，提高了系统的可靠性。

设备驱动框架层是对同类硬件设备驱动的抽象，将不同厂家的同类硬件设备驱动中相同的部分抽取出来，将不同部分留出接口，由驱动程序实现。

设备驱动层是一组驱使硬件设备工作的程序，实现访问硬件设备的功能。它负责创建和注册 I/O 设备，对于操作逻辑简单的设备，可以不经过设备驱动框架层，直接将设备注册到 I/O 设备管理器中，使用序列图如下图所示，主要有以下 2 点：

设备驱动根据设备模型定义，创建出具备硬件访问能力的设备实例，将该设备通过 bs\_device\_register() 接口注册到 I/O 设备管理器中。

应用程序通过 bs\_device\_find() 接口查找到设备，然后使用 I/O 设备管理接口来访问硬件。

## 框架分层介绍

### 2.3.1 核心层

为用户（应用层）提供抽象接口，将各层有机的连接起来，是整个框架的核心部分。应用程序通过该层提供的标准接口访问底层设备，设备驱动程序的升级、更替不会对上层应用产生影响。

这种方式使得设备的硬件操作相关的代码能够独立于应用程序而存在，双方只需关注各自的功能实现，从而降低了代码的耦合性、复杂性，提高了系统的可靠性。

### 2.3.2 片内\外驱动层

该层管理通用的片内外设驱动和片外外设驱动，片内外设比如pin、uart、iwd等，片外外设比如infrared、eeprom、阀门、各种传感器等。该层是抽象层，它将不同厂家的同类硬件设备驱动中相同的部分抽取出来，将不同部分留出接口，由驱动程序实现。

### 2.3.3应用层

该层为用户的应用代码，用户使用BARE SYSTEM提供的接口函数可直接开发逻辑业务，不需要再关心底层如何实现，之后只需关注并使用硬件的open、close、read、write、control这5个函数。因为使用的和硬件有关的代码都是标准化的接口，因此开发出来的应用可直接放入仓库维护。

### 2.3.4 bsp层

该层是驱使硬件设备工作的程序，实现访问硬件设备的功能。

它负责创建和注册 I/O 设备，并且使用mcu厂商提供的驱动代码实现某个外设的open、close、read、write、control功能。

应用程序通过 bs\_device\_find() 接口查找到设备，然后便可使用通用接口进行开发业务。

# 实现原理

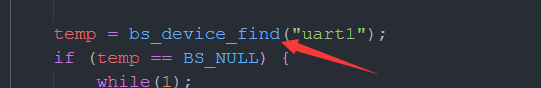
## 驱动框架设计思路

Bare system将任意芯片的的外设、板子的各种模块（红外、nb、eeprom）均看作一个个具体的设备，而操作这些设备均可抽象为register、open、close、read、write、control这六个行为。

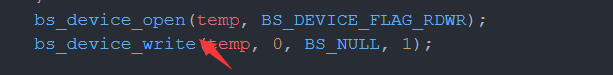
应用端的开发人员只需关注open、close、read、write、control这五个行为的**使用方法**，通过这五个抽象函数可以完成应用程序的业务开发，不用也无需关心这个设备底层硬件是什么。

而bsp端开发人员则相反，bsp端开发人员需根据具体的硬件型号完成最底层的open、close、read、write、control这五个行为的**实现**，然后将已经开发完成的具体型号的设备bsp驱动存入仓库，以后这个型号的设备均可直接进行业务开发。**等于开发一次bsp驱动可供所有人永久使用，而不用某一个工程师用一款芯片后无法和别人共享已经开发好的驱动。**

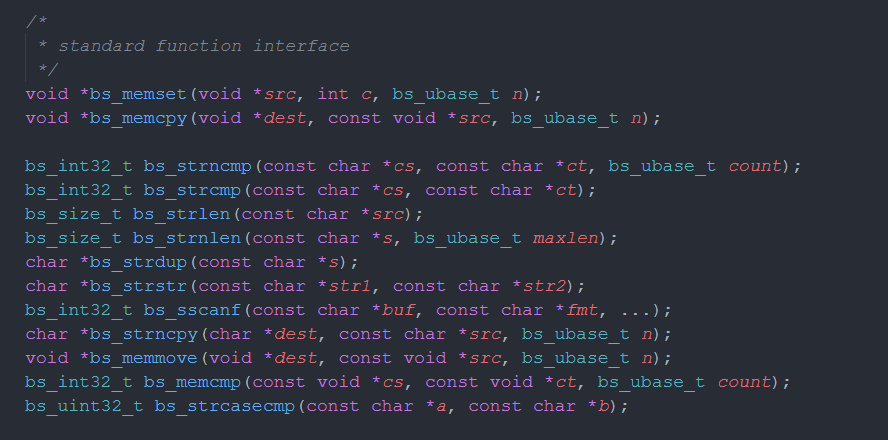
Bare system就是作为应用端和bsp驱动端的**架构桥梁**，应用端与bsp端均只依赖于Bare system。当bsp端将设备注册到Bare system后，Bare system使用链表机制管理这些已经注册好的驱动，应用端使用某个设备时调用函数bs\_device\_find查找这个设备，如下图;

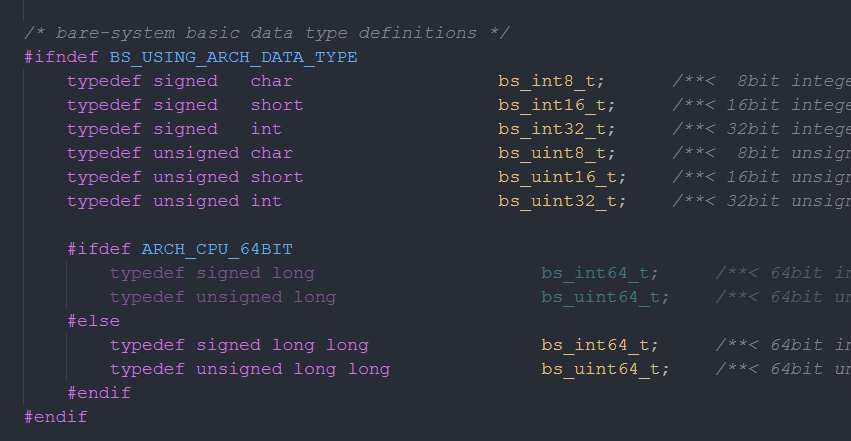


如果在链表中找到了这个设备，则可以使用该设备的句柄进行读写等各种操作。如下图：



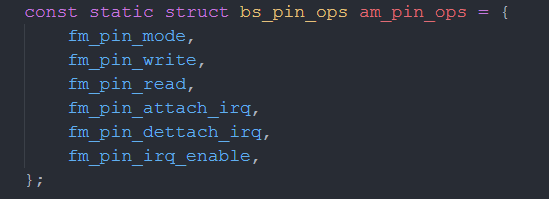
Bare system还提供应用端与bsp端通用的**库函数及各种类型定义**，以便于更彻底的解耦，如下图：





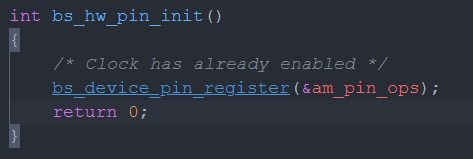
## PIN设备框架

Pin设备驱动框架较为简单，bsp端需要实现以下6个函数：



这6个函数需要根据具体的芯片型号调用厂家提供的库函数，或者操作寄存器完成。

实现完成后调用bs\_device\_pin\_register进行注册。

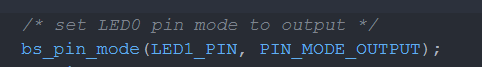


注册完成后即可投入使用。操作一个LED灯(PC0)例程如下：

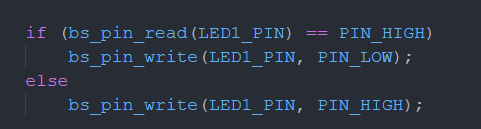
1. 用宏定义查找PC0的管教所在位置



1. 配置gpio模式



1. 调用读写函数即可



## SERIAL设备框架

## I2C设备框架

## CAN设备框架

## RTC设备框架

## ADC设备框架