**eBox配置相关**

**打造eBox生态圈**

**版本：版本19.3.05**

**发布日期：2019年3月**

本指南内容及产如有更新，请参考最新手

勘误记录

|  |  |  |
| --- | --- | --- |
| 日期 | 摘要 | 提交者 |
|  |  |  |
|  |  |  |

**如何使用本API手册**

本手册更新可能没有程序更新的快，部分内容可能有所书写错误，最终以代码中的内容为准。

如发现有文字性错误或者部分代码错误请在QQ群反馈139218853。

# 目录

[目录 1](#_Toc2791685)

[第1章 文件结构 2](#_Toc2791686)

[第2章 选择MCU 2](#_Toc2791687)

[2.1 F0平台（HAL固件库） 2](#_Toc2791688)

[2.2 F1平台（标准固件库） 3](#_Toc2791689)

[第3章 裁剪eBox 4](#_Toc2791690)

[3.1 #define USE\_PRINTF 4](#_Toc2791691)

[3.2 #define EBOX\_DEBUG 4](#_Toc2791692)

[3.3 #define DBG\_UART 5](#_Toc2791693)

[3.4 #define FLASH\_OVERRIDE 5](#_Toc2791694)

# 文件结构

# 选择MCU

Ebox中对于不同内核的MCU提供了不同的Project，分别以stm32f0xx，stm32f10xx，stm32f3xx，stm32f4xx命名，也就是说目前支持4个大的平台，后续增加也会采用类似的命名方式。

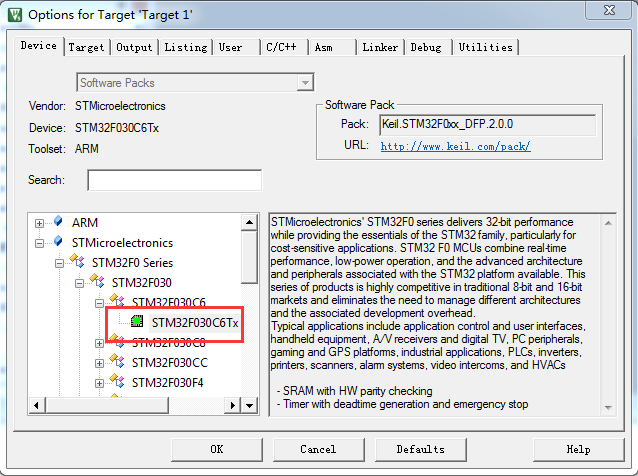
其中F1，F4使用了标准固件库，使用条件编译的方式初始话调用外设。

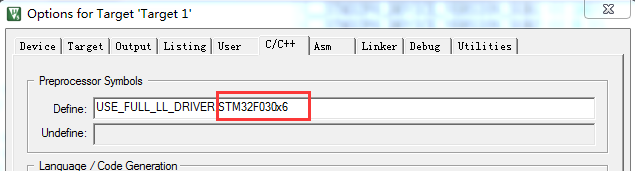
F0，F3使用了HAL固件库，使用查表的方式来初始化调用外设，外设定义在stm32fxxx\_define.h中。所以不同平台的配置稍有差异，发面分别以f0和f1平台为例说明

## F0平台（HAL固件库）

1. 第一步，在MDK的option中选择目标Device，并在C/C++选项卡中填写MCU名，

如果不确定该如何写可以在stm32f0xx.h文件中查找，确定保存

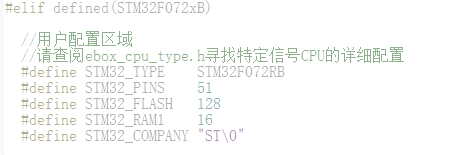




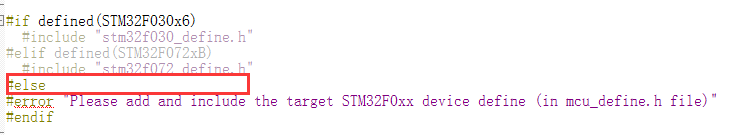
1. 点击编译，如果出现以下错误信息，请查看步骤3，如不报错，跳到步骤5

#35 "Please config target STM32F0xx device used in your application (in mcu\_config.h file)"

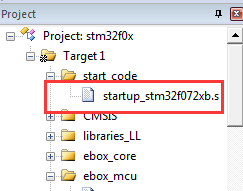
1. 打开mcu\_config.h文件，根据注释提示添加相应的MCU信息定义，类似下图



1. 参考stm32f030\_define.h文件，新建相应mcu的define文件，并在mcu\_define.h中添加引用

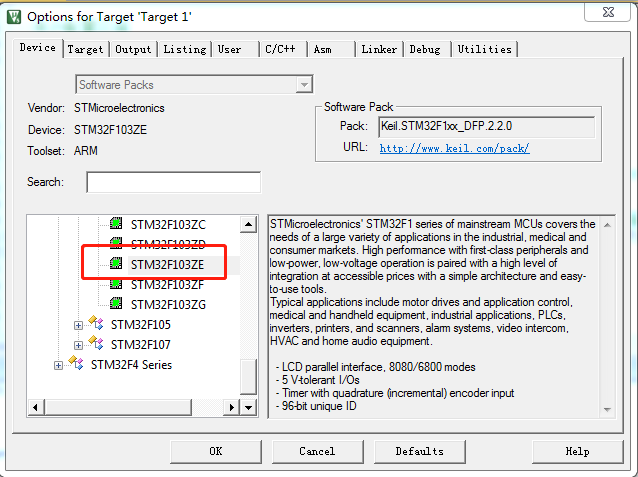


1. 此时，应该可以编译通过，如果有特殊外设要求，请替换Project->Target1->star\_code下的 .s文件

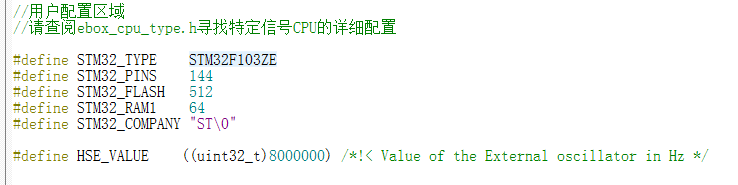


## F1平台（标准固件库）

1. 第一步，在MDK的option中选择目标Device，确定保存



1. 打开mcu\_config.h，配置下列信息



1. 在Project->Target1->CMSIS中选择对应的 .s文件，进行编译

# 裁剪eBox

目前eBox的配置项主要针对printf，其他配置后续会陆续添加，关于ebox的配置统一在ebox\_config.h中，改文件在user目录下

## #define USE\_PRINTF

定义使用哪种printf

1：使用标准库的printf，优点：功能齐全，可靠

2：使用自定义printf，优点：节省空间，可能有bug

同样条件下，分别选择1和2进行编译，空间占用情况如下：

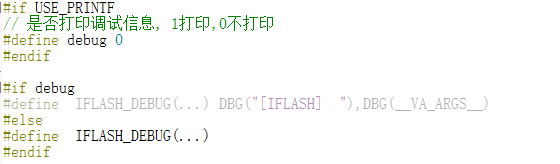
1. Program Size: Code=21856 RO-data=1892 RW-data=152 ZI-data=3096
2. Program Size: Code=21856 RO-data=1892 RW-data=152 ZI-data=3096

## #define EBOX\_DEBUG

是否使用print输出调试信息，如果USE\_PRINTF = 0或未定义则该配置无效

0：不使用； 1：使用

这个做为是否使用DEBUG输出信息的全局开关，实际上在每个模块内还有单独开关，类似下面的代码



## #define DBG\_UART

定义信息输出设备，该设备必须有write(\*buf,size)方法。默认uart1

当我们需要输出调试信息时通过 DBG(“调试信息…”)输出调试信息

## #define FLASH\_OVERRIDE

配置内部FLASH的写入方式

0：读->修改->擦除->写入，这种方式下会保存该区块的原始数据

1：擦除->写入，这种方式下直接写入，节省资源，但不会保存该区块原始数据