



Dev Cube

用户手册

版本: 1.0

版权 @ 2020

www.bouffalolab.com

1 Dev Cube 简介	4
2 镜像组成	5
3 IOT 程序下载	6
3.1 配置程序下载方式	6
3.2 配置下载参数	7
3.3 下载程序	8
4 MCU 程序下载	10
4.1 配置固件下载方式	10
4.2 配置镜像参数	11
4.3 配置高级镜像参数	11
4.4 下载程序	12
5 设置硬件安全参数	14
5.1 配置程序下载方式	15
5.2 配置密钥参数	15
6 Flash 调试助手	17
6.1 配置程序下载方式	18
6.2 读擦 Flash 内容	18
6.3 读写寄存器内容	19

List of Figures

1.1	Dev Cube 的芯片选择界面	4
2.1	启动镜像组成结构	5
3.1	IOT 程序下载方式选择界面	7
3.2	烧录文件选择界面	8
3.3	IOT 下载程序	8
3.4	hello world 程序运行效果	9
4.1	MCU 固件下载方式选择界面	10
4.2	镜像参数选择界面	11
4.3	高级镜像参数选择界面	12
4.4	下载程序	12
4.5	hello word 程序运行效果	13
5.1	硬件参数配置界面	14
5.2	下载方式界面	15
5.3	密钥参数配置界面	16
6.1	Flash 调试助手界面	17
6.2	下载方式界面	18
6.3	读擦 Flash 界面	19
6.4	读写寄存器界面	19

Dev Cube 是博流提供的芯片集成开发工具，包含 IOT 程序下载、MCU 程序下载和 RF 性能测试三种功能。本文档主要介绍 IOT 和 MCU 程序下载相关配置，RF 性能测试请参考《射频性能测试使用手册》。

Dev Cube 提供用户下载应用程序的功能，并且支持配置时钟、flash 等参数，可根据用户需求对程序进行加密、添加签名，还具备烧录用户资源文件、分区表等功能。

Dev Cube 的主要功能如下：

1. 支持 IOT 应用程序和 MCU 应用程序的下载
2. 支持多种型号 Flash 的擦、写、读
3. 支持各类文件下载到 Flash 并验证
4. 下载通讯接口支持 UART 和 JLink, 下载速度可配

用户可以通过 [Bouffalo Lab Dev Cube](#)，获取最新版本的 Dev Cube。双击解压后文件夹中的 BLDevCube.exe，在 Chip Selection 对话框中选择对应的芯片型号，点击 Finish 进入 Dev Cube 主界面。

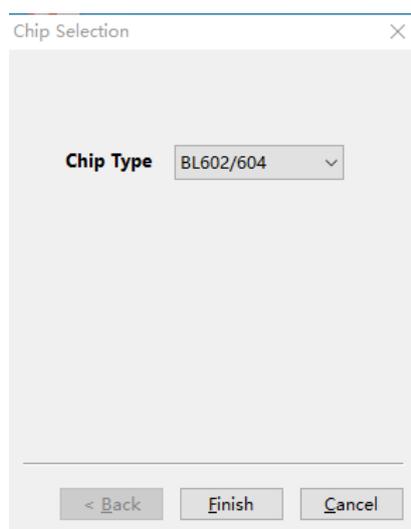


图 1.1: Dev Cube 的芯片选择界面

无论是 IOT 程序还是 MCU 程序，它们的镜像组成结构是相同的，均如下图所示：

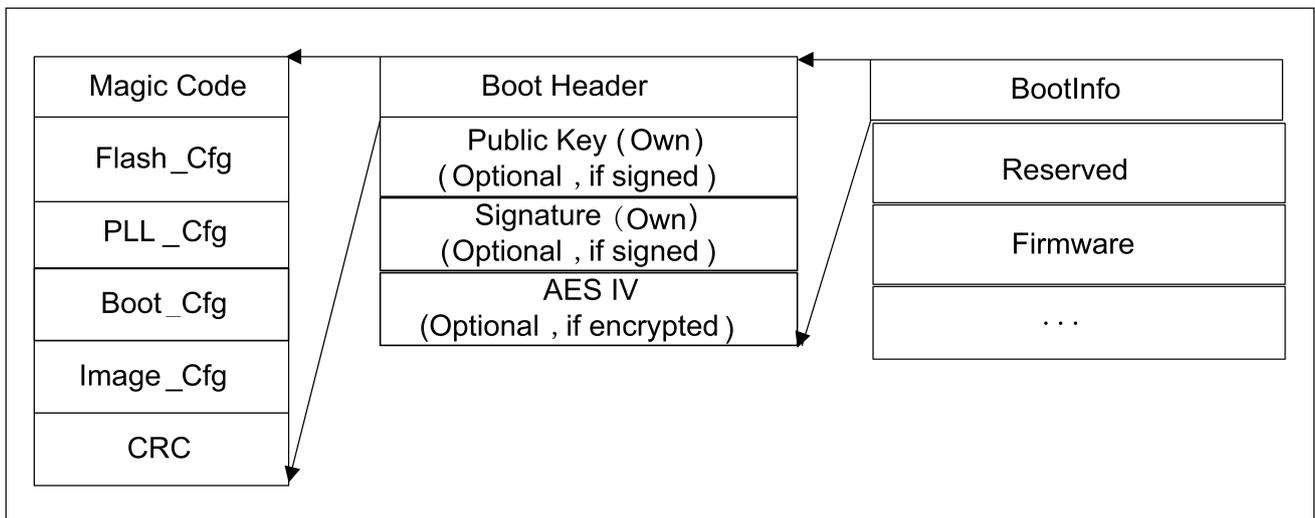


图 2.1: 启动镜像组成结构

启动镜像主要由两部分组成：

- **BootInfo** 主要包含 BootInfo 的 Magic Code, Flash 的配置信息, PLL 配置信息, 启动参数信息以及镜像配置信息等
- **Firmware** 固件, 即应用程序代码

如果只下载应用程序是无法使芯片正常工作的, 必须要将启动信息 **BootInfo** 下载到指定位置。以单核下载为例, 需要根据硬件电路的实际参数, 将 XTAL、PLL、Flash 等配置信息烧录到 **Bootinfo Addr** 对应的地址中, 将应用程序编译后的 bin 文件烧录到 **Image Addr** 对应的地址中。

在 **View** 菜单中选择 **IOT** 选项，会进入 IOT 程序下载界面，主要分为程序下载方式配置和烧录文件配置两部分。

3.1 配置程序下载方式

• 配置参数包括：

- **Interface:** 用于选择下载烧录的通信接口，可以选择 **Jlink** 或者 **UART**, 用户根据实际物理连接进行选择
- **COM Port:** 当选择 **UART** 进行下载的时候这里选择与芯片连接的 **COM** 口号，可以点击 **Refresh** 按钮进行 **COM** 号的刷新
- **Uart Speed:** 当选择 **UART** 进行下载的时候，填写波特率，推荐下载频率 **2M**
- **Board:** 选择所使用的板子型号, 板子型号和晶振类型, 共同决定了 **DTS** 文件, 也就是确定了板级硬件配置参数
- **Chip Erase:** 默认设置为 **False**, 下载时按照烧录地址和内容大小进行擦除, 选择 **True** 时, 在程序烧录之前会将 **Flash** 全部擦除
- **Xtal:** 用于选择板子所使用的晶振类型

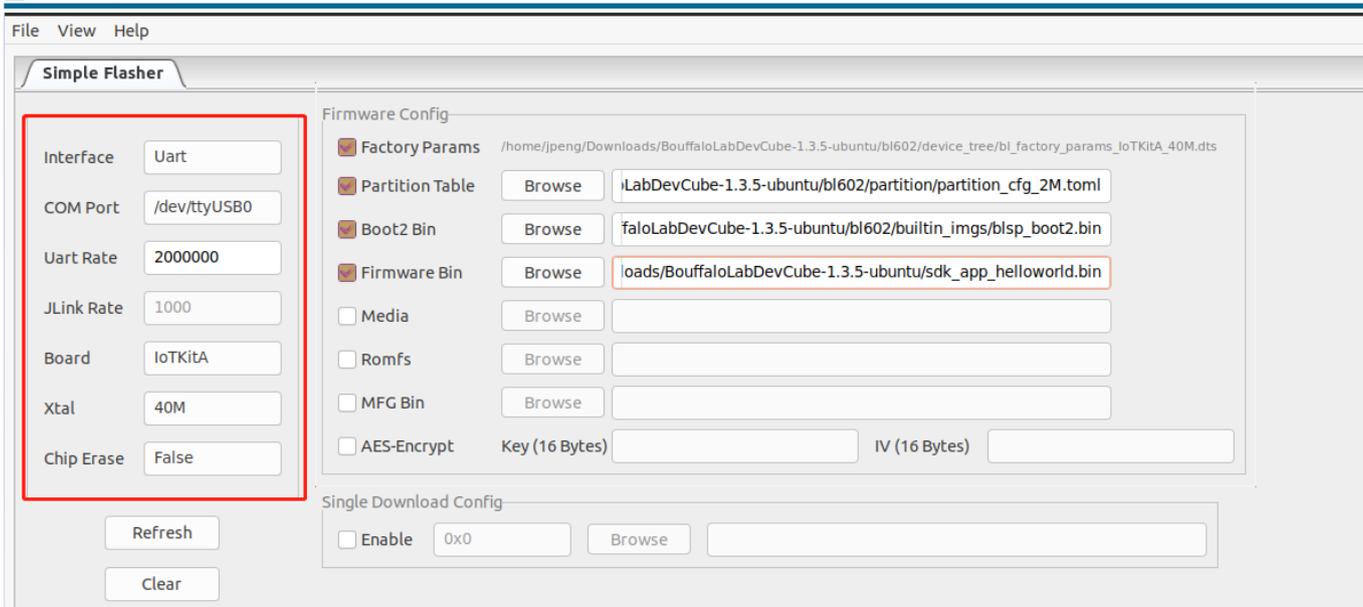


图 3.1: IOT 程序下载方式选择界面

3.2 配置下载文件

- 配置参数包括:

- **Partition Table:** 使用 Dev Cube 目录下对应芯片型号 `partition` 文件夹中的分区表, 分区文件主要是根据 Flash 大小确定, 默认选择 2M 的分区表配置文件
- **Boot2 Bin:** 它是系统启动后运行的第一个 Flash 程序, 负责建立 BLSP 安全环境, 并引导主程序运行, 使用 Dev Cube 目录下对应芯片型号 `builtin_imgs` 文件夹中的 Boot2 Bin 文件
- **Firmware Bin:** 用户编译生成的 bin 文件, 这里选择生成的 `helloworld.bin`
- **Media/Romfs:** Media 和 Romfs 二选一, 如果勾选 Media, 选择的是文件, 如果勾选 Romfs, 则选择的是文件夹
- **MFG Bin:** 选择 MFG 文件, MFG 文件是 RF 产测时候使用的应用程序, 根据晶振类型, 选择 Dev Cube 目录下对应芯片型号 `builtin_imgs/mfg` 文件夹中的 mfg bin 文件
- **AES-Encrypt:** 如果使用加密功能, 需要将 AES-Encrypt 选项选中, 并在旁边的文本框中输入加密所使用的 Key 和 IV。输入的是十六进制对应的“0”~“F”, 一个 Byte 由两个字符构成, 所以 Key 和 IV 分别要求输入 32 个字符。需要注意的是 IV 的最后 8 个字符 (即 4Bytes) 必须全为 0
- **Single Download Config:** 勾选 Enable 后可下载单个 Raw 文件到指定的 Flash 地址, 在左侧文本框填写下载的起始地址, 以 0x 打头

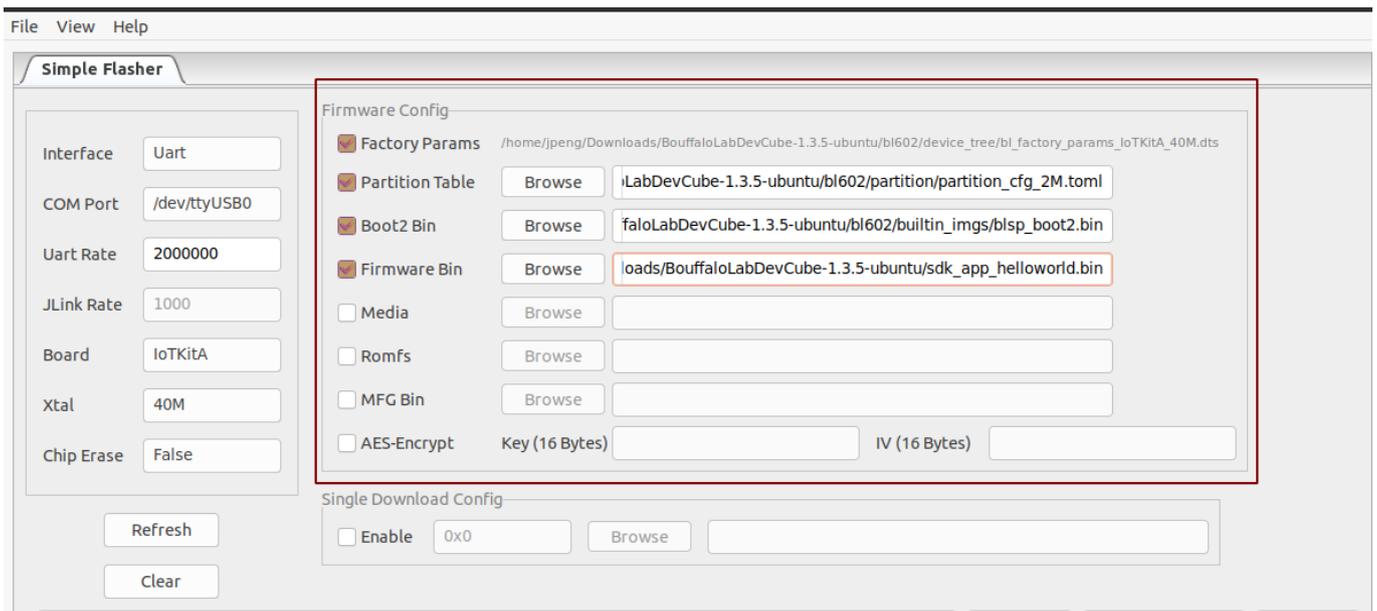


图 3.2: 烧录文件选择界面

3.3 下载程序

- 当选择 UART 方式烧录程序, 需要将板子的 BOOT 设置为高电平, 复位芯片, 使其处于 UART 引导下载的状态 (如果用户板子的 Boot 引脚和 Reset 引脚, 都与 USB 转串口的 DTR 和 RTS 连接, 则无需手动设置, 下载程序会自动设置 Boot 引脚和 Reset 芯片)。当选择 Jlink 方式烧录时, 可以一直将 Boot 引脚设置为低电平, 让其处于从 Flash 启动的状态
- 点击 Create&Download, 工具即可自动生成应用程序镜像和启动参数配置文件并开始烧写配置的各个文件。出现下图 log 信息, 程序下载成功

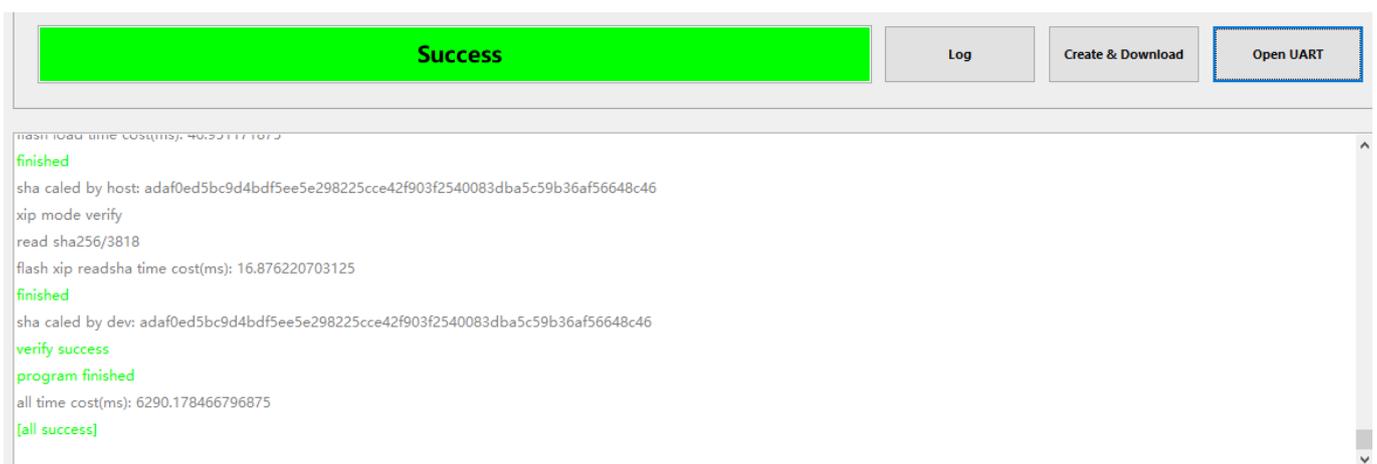
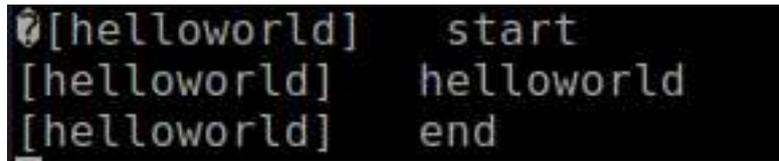


图 3.3: IOT 下载程序

注解：若没有连接板子，只需生成应用程序镜像和启动参数配置文件，也是点击 **Create&Program** 按钮

- 下载成功后，将板子的 BOOT 引脚设置为低电平，复位芯片，使其从 Flash 启动, 此时, 应用程序即可运行起来

下图是 **hello world** 程序运行起来的效果。



```
[helloworld] start
[helloworld] helloworld
[helloworld] end
```

图 3.4: hello world 程序运行效果

在 View 菜单中选择 MCU 选项，会进入 MCU 程序下载界面，主要分为固件下载方式配置、镜像参数配置和高级镜像参数配置三个主要部分。

4.1 配置固件下载方式

• 配置参数包括：

- Interface: 用于选择下载烧录的通信接口，可以选择 Jlink 或者 UART, 用户根据实际物理连接进行选择
- COM Port: 当选择 UART 进行下载的时候这里选择与芯片连接的 COM 口号，可以点击 Refresh 按钮进行 COM 号的刷新
- Uart Speed: 当选择 UART 进行下载的时候，填写波特率，推荐下载频率 2M
- Chip Erase: 默认设置为 False，下载时按照烧录地址和内容大小进行擦除，选择 True 时，在程序烧录之前会将 Flash 全部擦除
- Xtal: 用于选择板子所使用的晶振类型

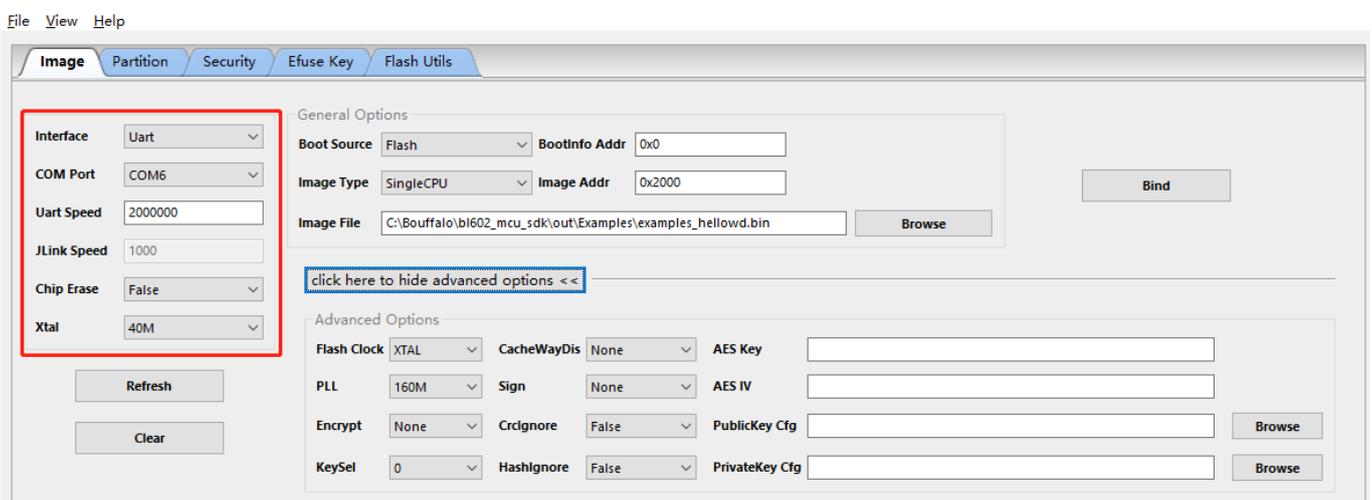


图 4.1: MCU 固件下载方式选择界面

4.2 配置镜像参数

- 配置参数包括：
 - **Boot Source:** 默认为 Flash, 只有在需要生成从 UART 或者 SDIO 启动镜像的时候才需要选择 UART/SDIO
 - **BootInfo Addr:** Bootinfo 启动参数的存放地址, 对于单核程序, 填写 0x0, 对于双核的 CPU0 镜像, 填写 0x0, 对于双核的 CPU1 镜像, 填写 0x1000
 - **Image Type:** SingleCPU 用于生成单核的镜像, CPU0 用于生成双核中 CPU0 的镜像, CPU1 用于生成双核中 CPU1 的镜像, Boot2 用于生成 Boot2 镜像, RAW 用于下载用户自定义的原始资源文件
 - **Image Addr:** 应用程序的存放地址, 建议填写 0x2000 或者 0x2000 以后的地址
 - **Image File:** 选择应用程序的 Bin 文件或者用户的资源文件

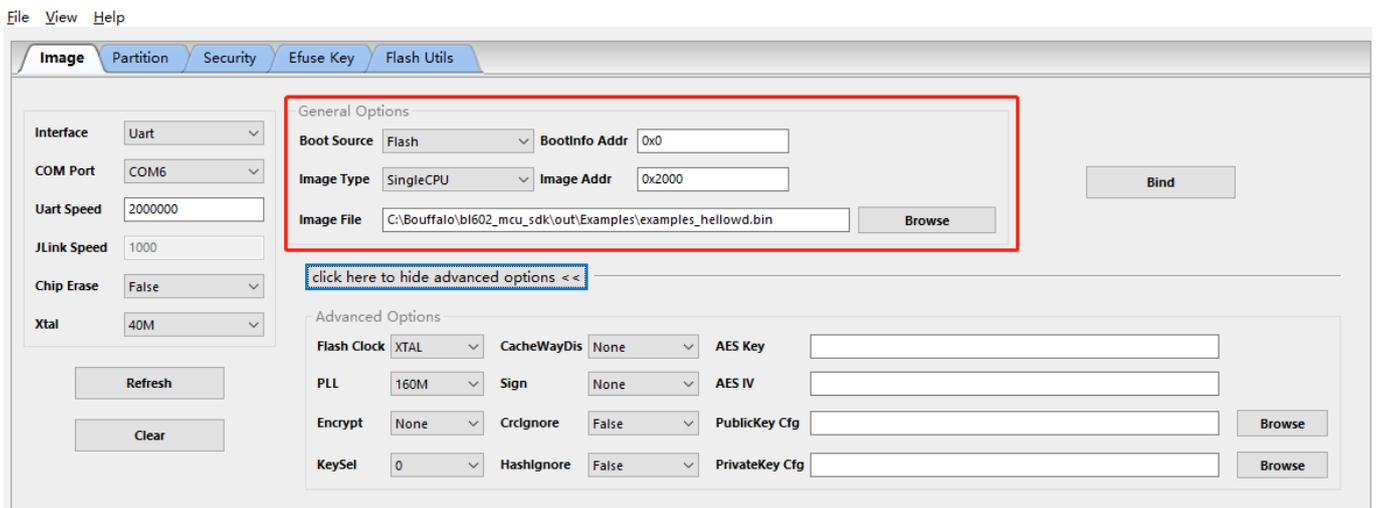


图 4.2: 镜像参数选择界面

4.3 配置高级镜像参数

- 当点击 `click here to show advanced options` 时, 会展开高级镜像配置, 可配置的参数包括：
 - **Flash Clock:** 用于设定 Flash 的时钟
 - **PLL :** PLL 时钟配置, 默认为 160M
 - **CacheWayDis :** L1C Cache 的 4 条 way 设定, 默认为 none, 即使能全部的 4 条 way
 - **Sign :** 选择是否对程序镜像进行 ECC 签名
 - **Crclgnore :** 是否需要 CRC 校验。当参数选择 False 时, 开启 Boot Info 的 CRC 校验; 反之, 不设置 Boot Info 的 CRC 校验
 - **Hashlgnore :** 是否需要做镜像的完整性 Hash 校验。当参数选择 False 时需要做 Hash 校验; 反之, 不做镜像的完整性校验

- **Encrypt** : 选择加密方式, 使能对程序镜像的加密功能. 使能加密功能后, 需要根据 AES 加密方式在 AES Key 和 AES IV 中输入对应的数值

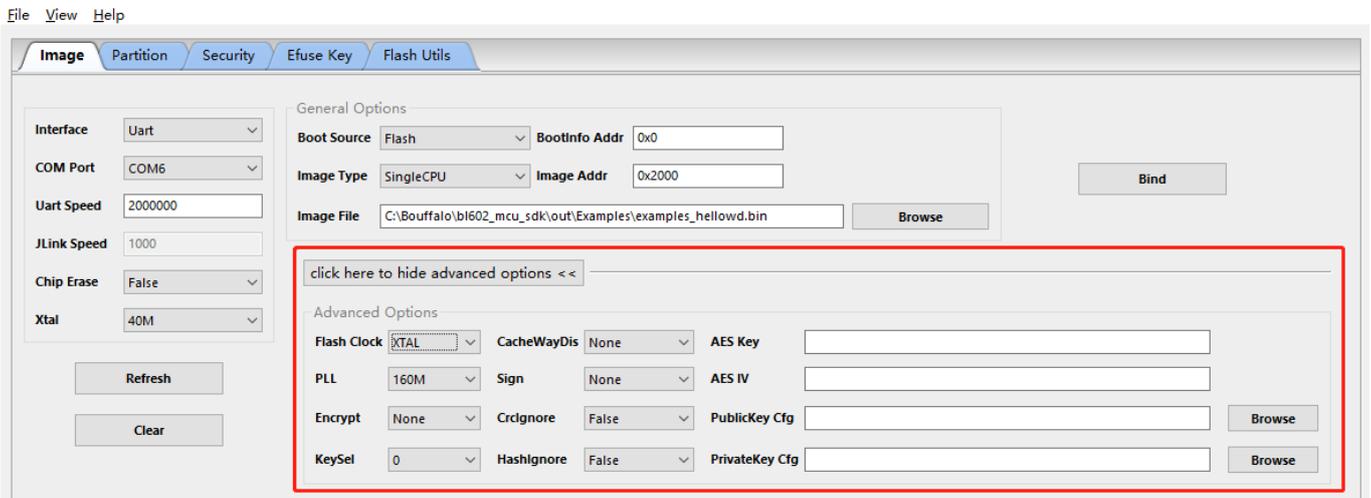


图 4.3: 高级镜像参数选择界面

4.4 下载程序

- 当选择 UART 方式烧录程序, 需要将板子的 BOOT 设置为高电平, 复位芯片, 使其处于 UART 引导下载的状态 (如果用户板子的 Boot 引脚和 Reset 引脚, 都与 USB 转串口的 DTR 和 RTS 连接, 则无需手动设置, 下载程序会自动设置 Boot 引脚和 Reset 芯片)。当选择 Jlink 方式烧录时, 可以一直将 Boot 引脚设置为低电平, 让其处于从 Flash 启动的状态
- 点击 **Create&Program**, 工具即可自动生成应用程序镜像和启动参数配置文件并开始烧写。出现下图 log 信息, 程序下载成功

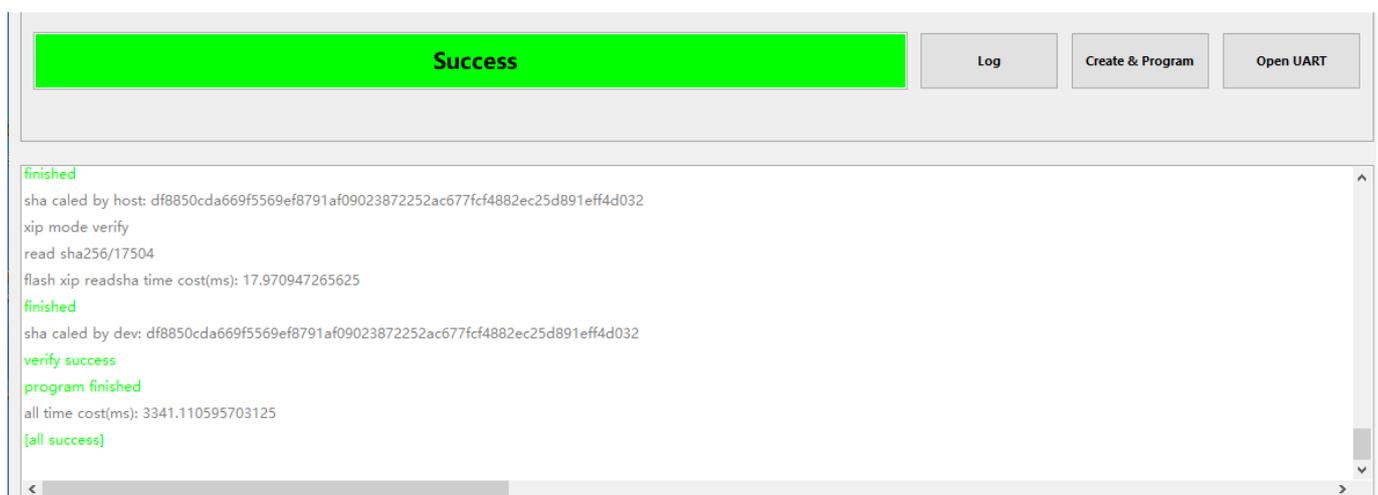


图 4.4: 下载程序

注解：若没有连接板子，只需生成应用程序镜像和启动参数配置文件，也是点击 **Create&Program** 按钮

- 下载成功后，将板子的 **BOOT** 引脚设置为低电平，复位芯片，使其从 **Flash** 启动，此时，应用程序即可运行起来

下图是 **hello word** 程序运行起来的效果。

```
system clock=160M
hello world
Loop 1000,1000
Loop 2000,1000
Loop 3000,1000
Loop 4000,1000
clic_timer_handlerLoop 5000,1000
```

图 4.5: hello word 程序运行效果

设置硬件安全参数

在 View 菜单中选择 MCU 选项，点击 Security 选项，进入硬件安全参数配置界面。配置信息包含固件下载方式配置、AES 模式配置和密钥配置。

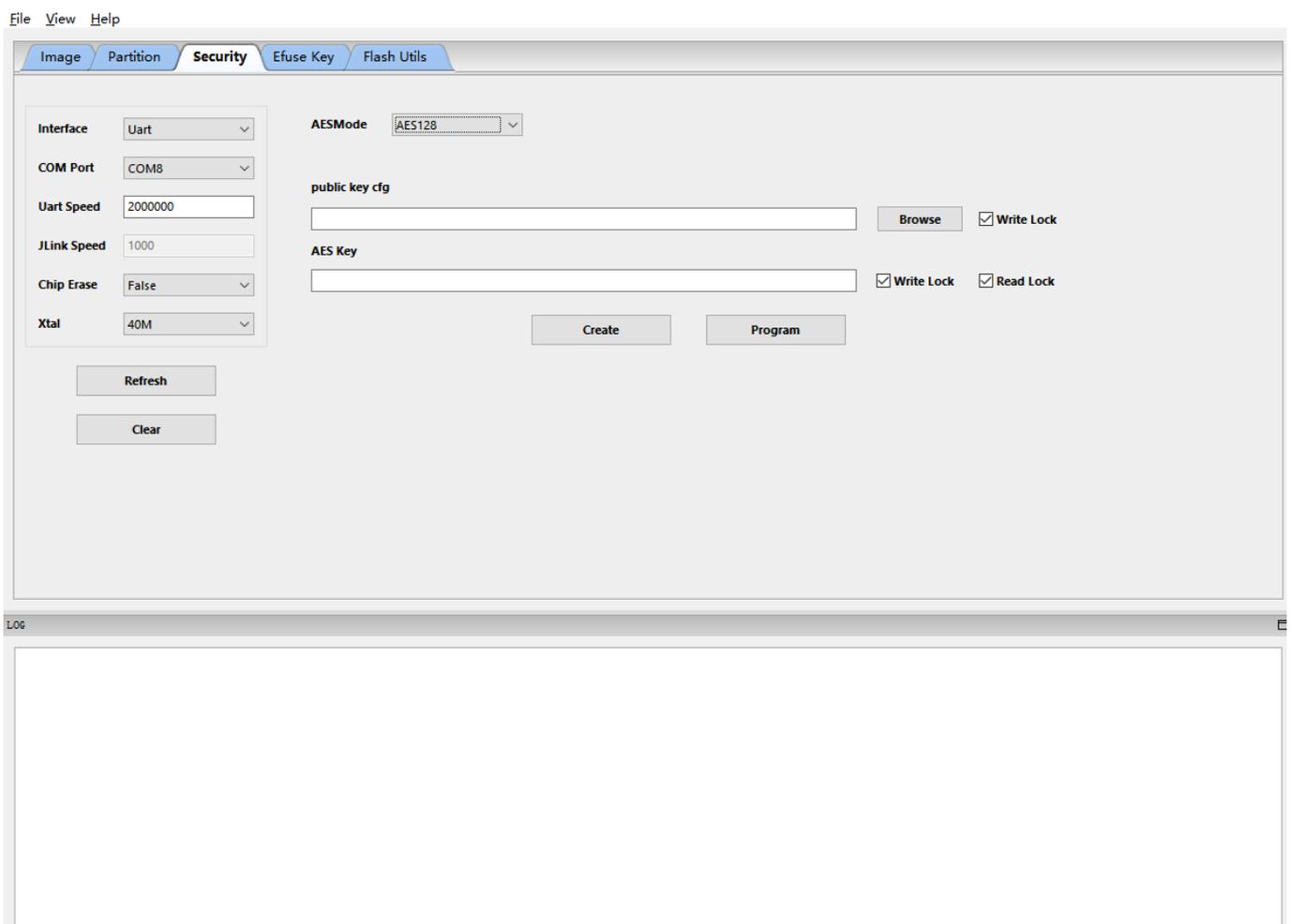
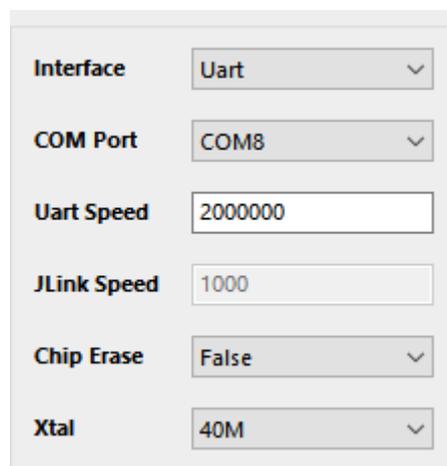


图 5.1: 硬件参数配置界面

5.1 配置程序下载方式

- 配置参数包括：
 - **Interface:** 用于选择下载烧录的通信接口，可以选择 **Jlink** 或者 **UART**, 用户根据实际物理连接进行选择
 - **COM Port:** 当选择 **UART** 进行下载的时候这里选择与芯片连接的 **COM** 口号，可以点击 **Refresh** 按钮进行 **COM** 号的刷新
 - **Uart Speed:** 当选择 **UART** 进行下载的时候，填写波特率，推荐下载频率 **2M**
 - **Chip Erase:** 默认设置为 **False**，下载时按照烧录地址和内容大小进行擦除，选择 **True** 时，在程序烧录之前会将 **Flash** 全部擦除
 - **Xtal:** 用于选择板子所使用的晶振类型



Interface	Uart
COM Port	COM8
Uart Speed	2000000
JLink Speed	1000
Chip Erase	False
Xtal	40M

图 5.2: 下载方式界面

5.2 配置密钥参数

若想要给芯片加密，除了在下载程序时，使用 **Image** 功能中的 **AES** 对芯片进行软件加密，还需要进行硬件加密。

- 在 **AESMode** 中选择对应的加密模式，本例中选择 **AES128**
- 在 **public key cfg** 中选择公钥的 **PEM** 文件，本例选择 **/common/pem/publickey_uecc.pem**
- **AES Key** 填写和软件加密相同的值

点击 **Create**，生成 **Efuse** 文件，点击 **Program**，烧录 **Efuse** 文件。

AESMode

public key cfg

Write Lock

AES Key

Write Lock Read Lock

图 5.3: 密钥参数配置界面

在 View 菜单中选择 MCU 选项，点击 Flash Utils 选项，进入 Flash 调试助手界面。Flash 调试助手用来获取 Flash ID、读取和擦除 Flash 中指定地址的内容、读取和写入对应寄存器的值。

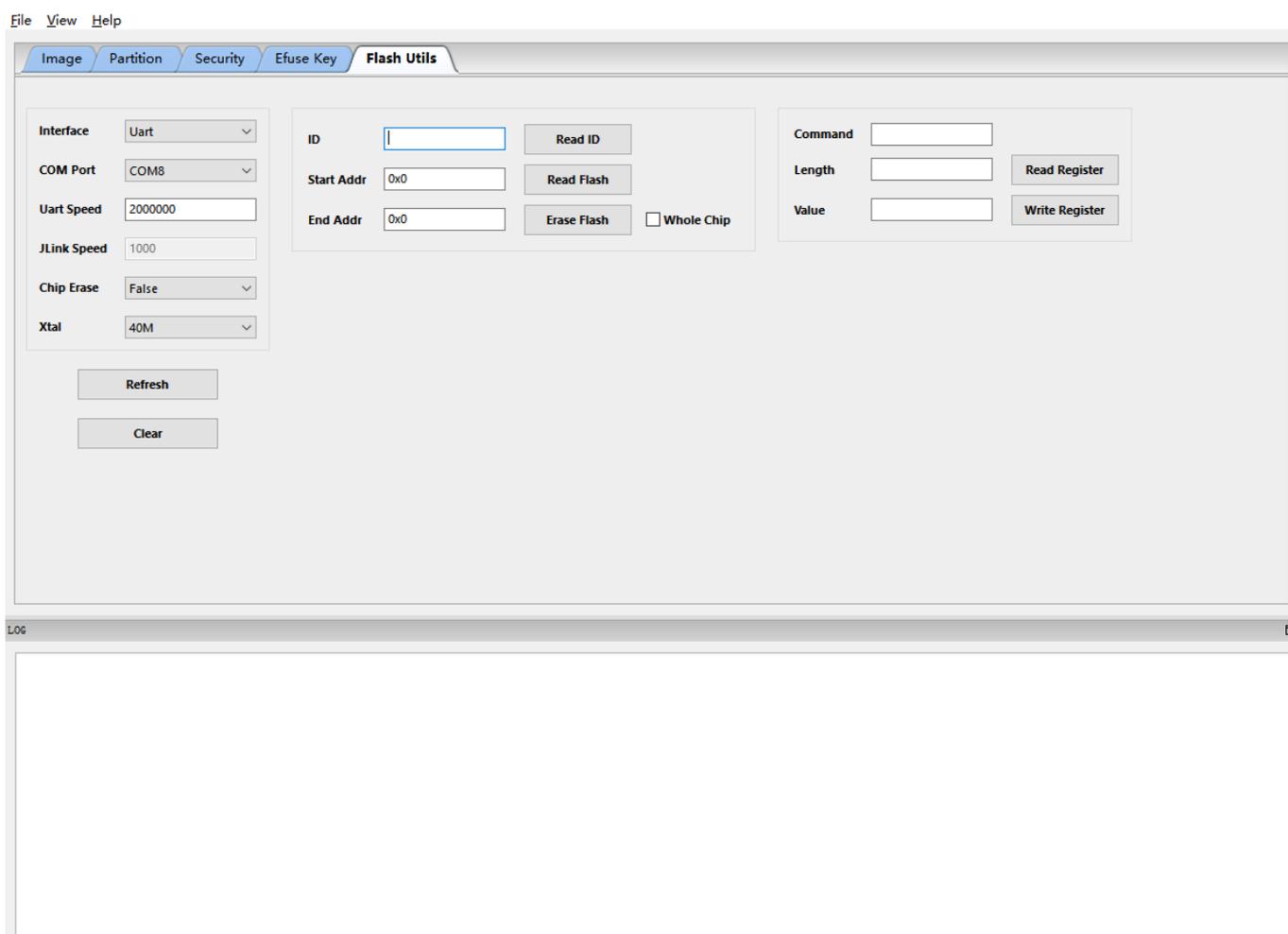
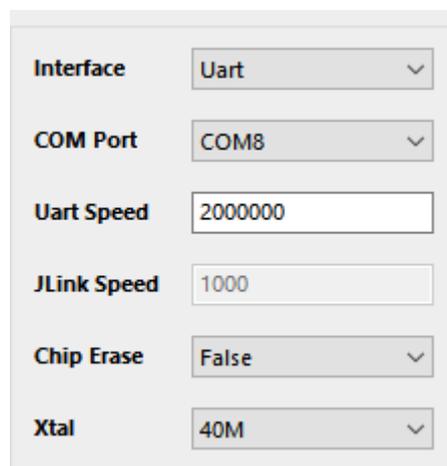


图 6.1: Flash 调试助手界面

6.1 配置程序下载方式

- 配置参数包括：
 - **Interface:** 用于选择下载烧录的通信接口，可以选择 **Jlink** 或者 **UART**, 用户根据实际物理连接进行选择
 - **COM Port:** 当选择 **UART** 进行下载的时候这里选择与芯片连接的 **COM** 口号，可以点击 **Refresh** 按钮进行 **COM** 号的刷新
 - **Uart Speed:** 当选择 **UART** 进行下载的时候，填写波特率，推荐下载频率 **2M**
 - **Chip Erase:** 默认设置为 **False**，下载时按照烧录地址和内容大小进行擦除，选择 **True** 时，在程序烧录之前会将 **Flash** 全部擦除
 - **Xtal:** 用于选择板子所使用的晶振类型

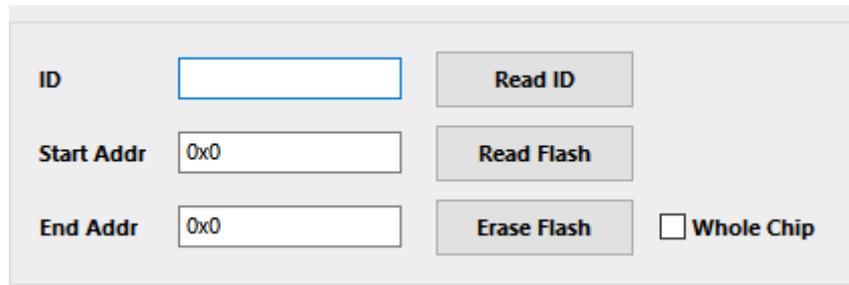


Interface	Uart
COM Port	COM8
Uart Speed	2000000
JLink Speed	1000
Chip Erase	False
Xtal	40M

图 6.2: 下载方式界面

6.2 读擦 Flash 内容

- 读取 Flash 的 ID: 点击 **Read ID**
- 读取 Flash 固定长度的值: 在 **Start Addr** 中设置需要读取数据的开始地址; 在 **End Addr** 中设置需要读取数据的结束地址, 点击 **Read Flash**, 读取的内容会存放在 **flash.bin** 文件中, 该文件路径为: **BouffaloLabDevCube-1.3.4-win32/flash.bin**
- 擦除 Flash 固定长度的值: 在 **Start Addr** 中设置需要擦除数据的开始地址; 在 **End Addr** 中设置需要擦除数据的结束地址, 点击 **Erase Flash**(若需要擦除整块芯片的值, 只需勾选 **Whole Chip**)

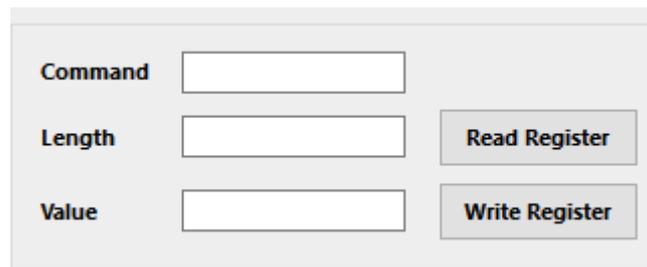


The interface for reading and erasing Flash memory. It features three input fields on the left: 'ID', 'Start Addr', and 'End Addr'. The 'Start Addr' and 'End Addr' fields are pre-filled with '0x0'. To the right of these fields are three buttons: 'Read ID' (aligned with ID), 'Read Flash' (aligned with Start Addr), and 'Erase Flash' (aligned with End Addr). A checkbox labeled 'Whole Chip' is located to the right of the 'Erase Flash' button.

图 6.3: 读擦 Flash 界面

6.3 读写寄存器内容

- 读取寄存器的内容: 在 Command 中输入读取命令 0x05/0x35, Length 中填写需要读取的位数, 点击 Read Register, 读取的数据显示在 Value 中
- 写入寄存器的内容: 在 Command 中输入写命令 0x01, Length 中填写需要写入的位数, 将写入的数据填写在 Value 中, 点击 Write Register



The interface for reading and writing registers. It features three input fields on the left: 'Command', 'Length', and 'Value'. To the right of these fields are two buttons: 'Read Register' (aligned with Length) and 'Write Register' (aligned with Value).

图 6.4: 读写寄存器界面