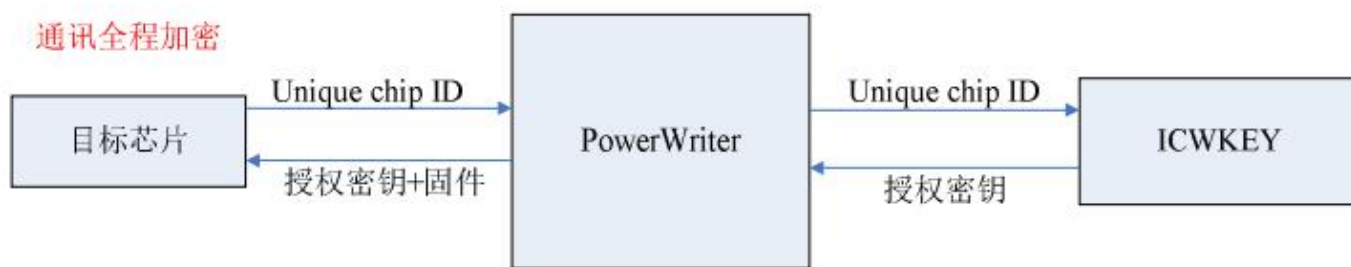




简介

创芯工坊致力于以互联网+技术赋能传统集成电路行业，为传统集成电路生产流程提供更为安全高效的安全生产管理模式！为了满足用户对于芯片安全烧录、授权控制以及定制化生产的需求，创芯工坊推出了“安全授权盾 (ICWKEY)”，以下简称为 ICWKEY，作为创芯工坊烧录器 PowerWriter 离线授权的一种辅助工具，负责控制授权次数和生成授权密钥，可以确保生产时，目标芯片+PowerWriter+ICWKEY 整个链路层数据的安全，确保用户的固件不被非法访问，确保用户手上保留唯一的授权控制权限，防止未经授权拷贝的可能，ICWKEY 完全掌握在用户手中，安全可靠，下图是工作流程示意图：



ICWKEY 提供了向量矩阵加密 (Matrix) 和椭圆曲线数字签名 (ECDSA) 两种 UID (Unique Chip ID) 授权算法，也提供 SDK 供用户开发自定义授权算法，以满足用户的不同需求。我们提供了目标芯片如何使用 UID 授权算法的范例程序，也提供了 ICWKEY 的 Windows 软件 ICWKEY.exe，用户可以将 ICWKEY.exe 随机生成的授权算法源代码导入到自己的程序。

用户在阅读本文档时需注意以下约定：

本文档的示例目标芯片为 STM32F103RG，用户可在实际使用中灵活调整；

示例中出现的相对路径都是相对于 ICWKEY.exe 软件的默认安装路径；

本文档的功能可能在产品更新迭代中，和实际产品会存在一定的出入，请以实际产品为准。

目录

1 快速使用指南.....	3
1.1 USB 驱动安装.....	3
1.2 设置项目名称和通讯密钥.....	4
1.3 生成 UID 授权算法.....	6
1.4 目标程序加入 UID 算法重新生成固件.....	9
1.5 PowerWriter 保存项目.....	11
1.6 离线烧录目标芯片.....	12
1.7 验证烧录.....	12
2 安全机制.....	12
2.1 ICWKEY 自身的安全性.....	12
2.2 UID 向量矩阵加密(Matrix)介绍.....	13
2.3 UID 椭圆曲线数字签名(ECDSA)介绍.....	14
2.4 如何开发自定义加密算法.....	14
2.5 安全优化的 UID 授权使用范例.....	15
3 软件功能详细介绍.....	16
3.1 菜单.....	16
1. 文件.....	16
2. 执行.....	17
3. 帮助.....	17
4. 语言.....	17
3.2 通讯设定.....	17
3.3 项目配置->设备配置.....	17
3.4 项目配置->UID 算法.....	18
3.5 项目配置->日志信息.....	19
4 UID 授权目标芯片使用范例.....	19
4.1 向量矩阵加密(Matrix)目标芯片程序范例.....	19
4.2 椭圆曲线数字签名(ECDSA)目标芯片程序范例.....	20
5 注意事项.....	21
5.1 导出的 UID 授权算法源代码必须根据目标芯片进行修改.....	21
5.2 务必保管好项目文件和密码.....	23
5.3 UID 授权在芯片中的存储位置不能和用户代码区重叠.....	23
6 常见问题.....	26
6.1 USB 驱动安装不成功.....	26
6.2 如何判断 PowerWriter 和 ICWKEY 连接成功.....	26
6.3 如何确认数据已经正确烧录到目标芯片.....	26
6.4 为什么加入了 UID 校验, 程序调式时不能通过校验.....	26
6.5 程序应该预留多大空间存放 UID 授权密钥.....	26
6.6 烧录失败的原因.....	26
6.7 烧录后二次回读或校验失败.....	27
7 联系我们.....	27
8 版本历史.....	28

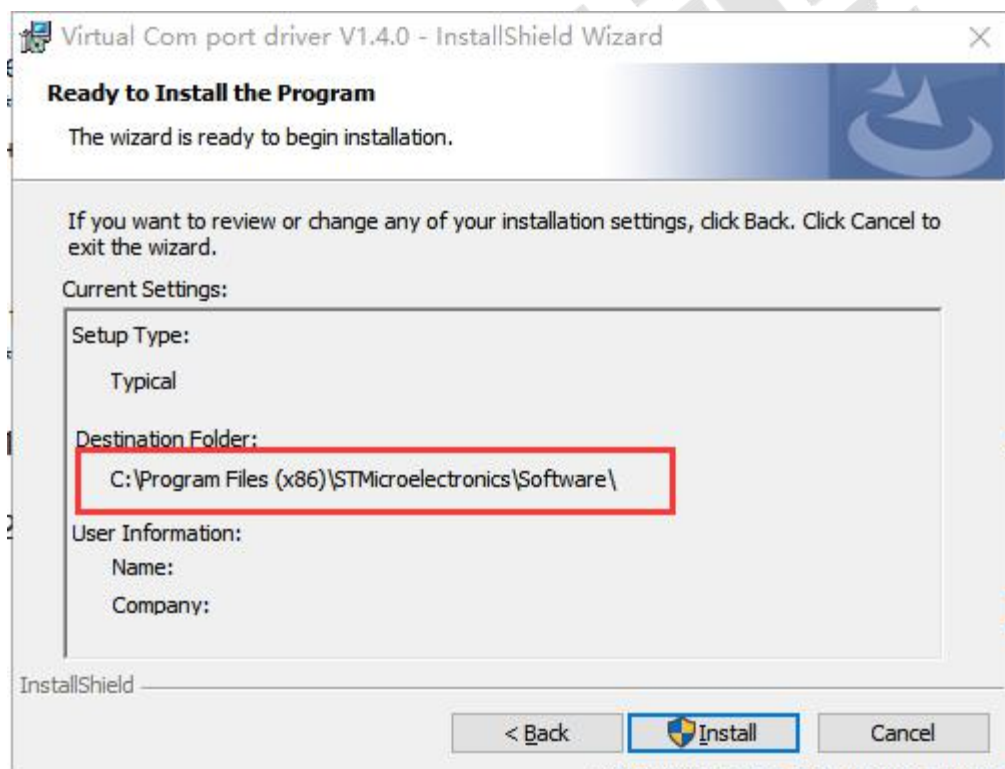
1 快速使用指南

1.1 USB 驱动安装

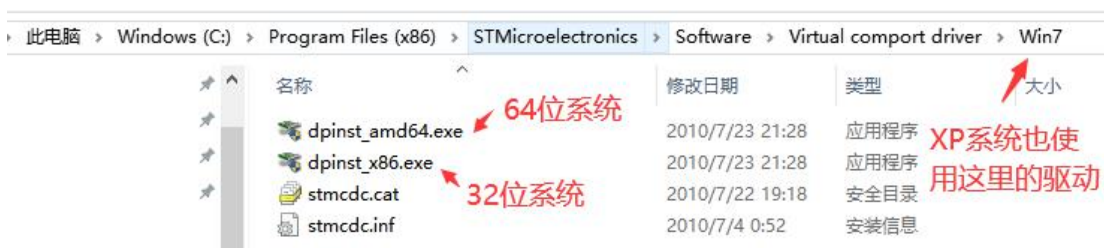
ICWKEY 使用 USB 虚拟串口 (Virtual COM Port) 和电脑连接，首次连接到电脑，需要安装驱动，如果电脑是 Win10 系统，系统会自动完成驱动安装，如果是比 Win10 更早的系统需要手动安装，安装包在 USB driver\STSW_STM32102_V1.4.0，先阅读 readme.txt，然后双击 VCP_V1.4.0_Setup.exe 启动解压，安装包将被解压到：

C:\Program Files (x86)\STMicroelectronics\Software\Virtual comport driver，根据电脑系统以管理员模式启动安装包：

Step1) 解压



Step2) 启动安装



Step3) 安装成功



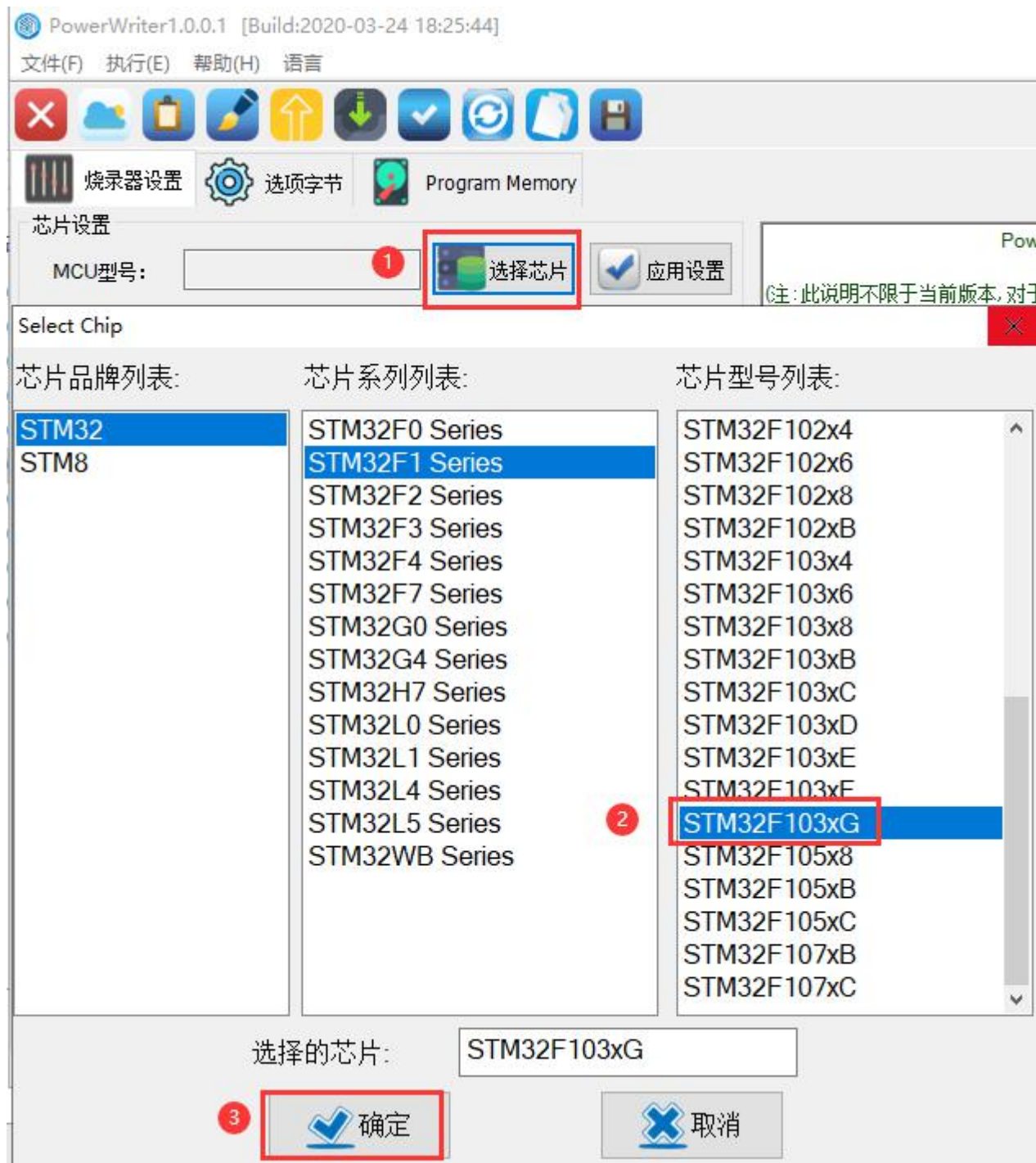
通过电脑设备管理器可以找到安装成功的设备。



1.2 设置项目名称和通讯密钥

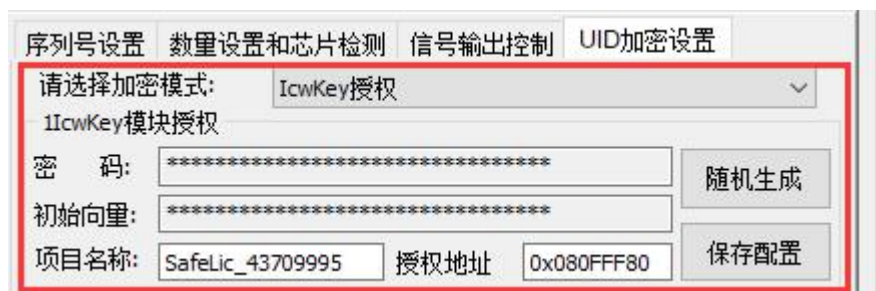
ICWKEY 需要和 PowerWriter 配合使用，两者必须使用相同的项目名称和通讯密钥才能完成通讯：

Step1) 在 PowerWriter.exe 选择目标芯片型号



根据目标芯片选择正确的型号，此处仅是示例

Step2) 设置项目名称和通讯密钥(包括密码和初始向量)



Step3) 修改授权地址



修改授权地址：0x080FF80=>0x0800FF80;

Step4) 将项目名称和通讯密钥复制到 ICWKEY. exe 项目配置界面



- ①连接一个 ICWKEY 设备;
- ②将 PowerWriter 设置的项目名称和通讯密钥复制到 ICWKEY 项目配置窗口;
- ③设置授权次数;
- ④设置 ICWKEY 可配置次数(非常重要), 用户可以决定 ICWKEY 是一次性使用还是可以重复多次使用。

1.3 生成 UID 授权算法

ICWKEY 支持了两种加密算法：向量矩阵加密 (Matrix) 和椭圆曲线数字签名 (ECDSA)

前者是通过将目标芯片的 Unique Chip ID 与一组用户设定 Key 进行算术运算和位操作，计算出另外一组 Key，

运算负荷和资源开销极小，后者是一种非对称加密算法，破解难度极大，但是运算负荷和资源开销较大，下面以向量矩阵加密为例：



①随即生成矩阵，未保存的项目，每次打开此对话框都会随机生成矩阵，已保存项目则不会；

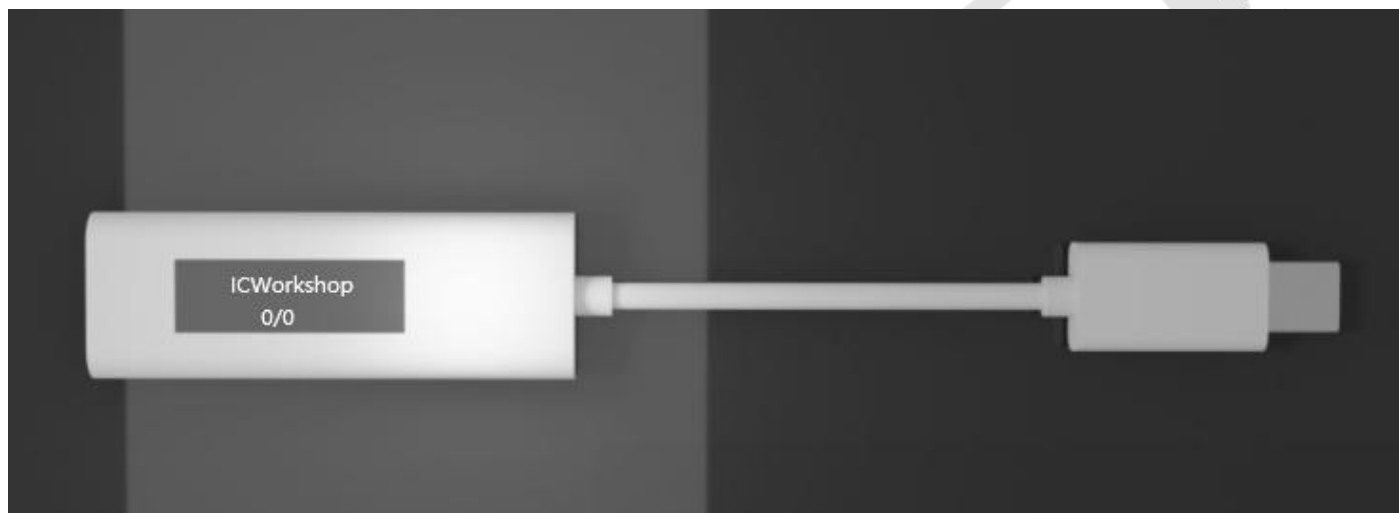
②保存矩阵。

保存 UID 算法后，将项目内容下载到 ICWKEY，新建项目将弹出保存对话框，**请妥善保管密码，否则无法再次打开项目文件**





下载前的 ICWKEY



下载后的 ICWKEY



1.4 目标程序加入 UID 算法重新生成固件

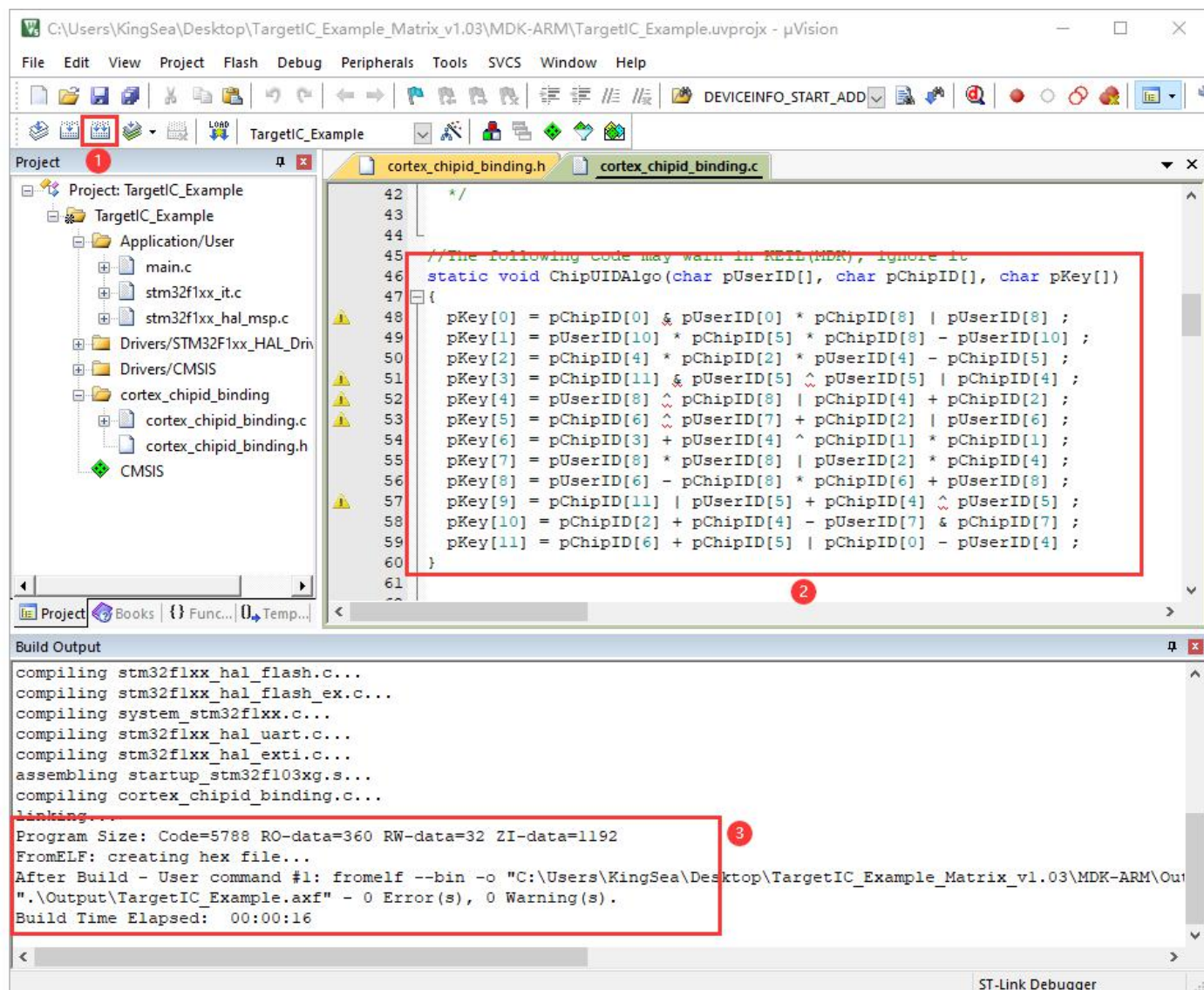
重新回到 UID 算法编辑器，导出 UID 加密源代码(cortex_chipid_binding.h&cortex_chipid_binding.c)，保存到用户的开发项目，这里使用的是示例程序，路径在 TargetIC_Example\TargetIC_Example_Matrix_v1.03



重新编译生成新固件(芯片烧录档案)，示例固件为：

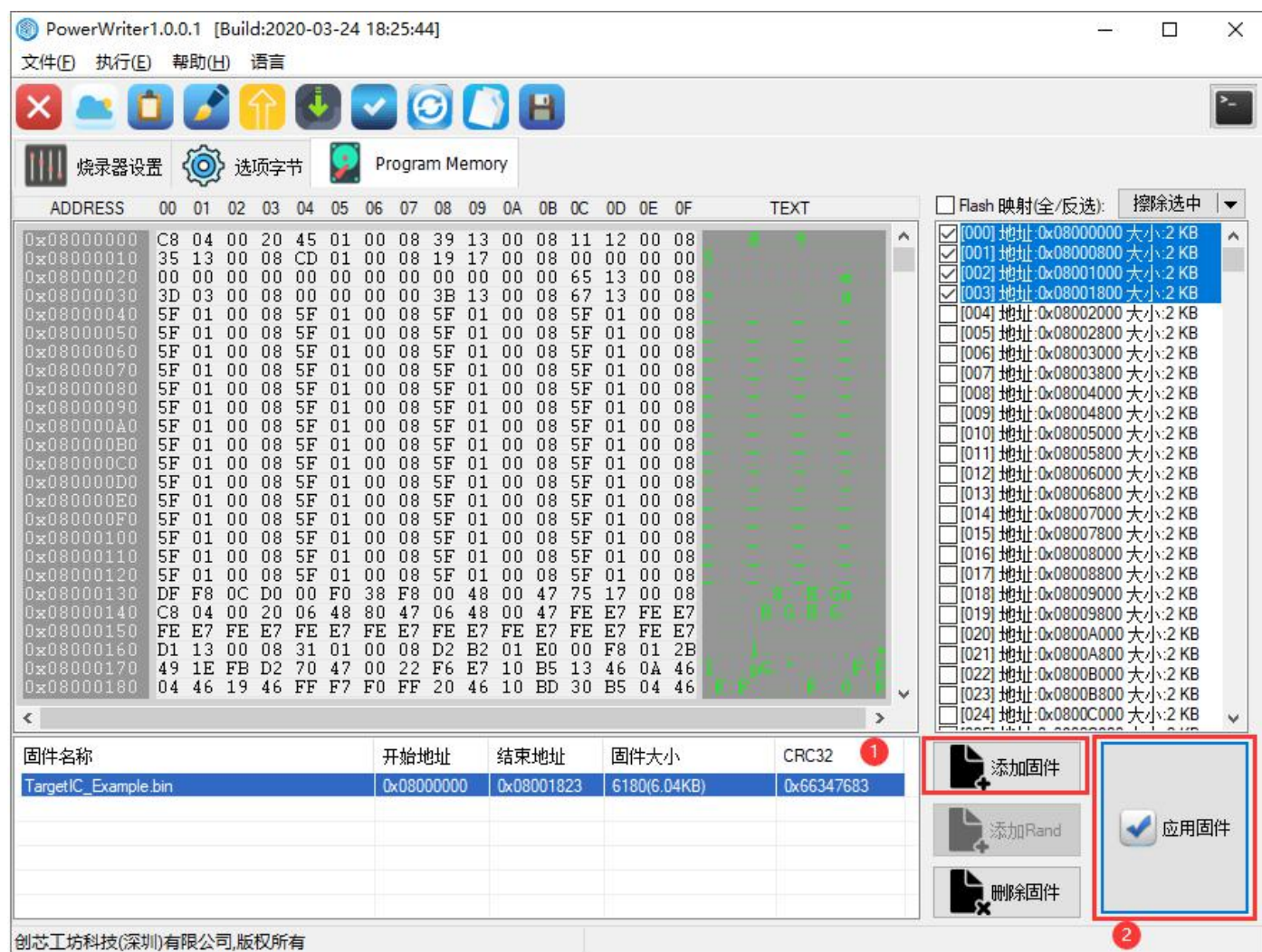
TargetIC_Example\TargetIC_Example_Matrix_v1.03\MDK-ARM\TargetIC_Example\TargetIC_Example.bin,

用户也可以使用 TargetIC_Example.hex;



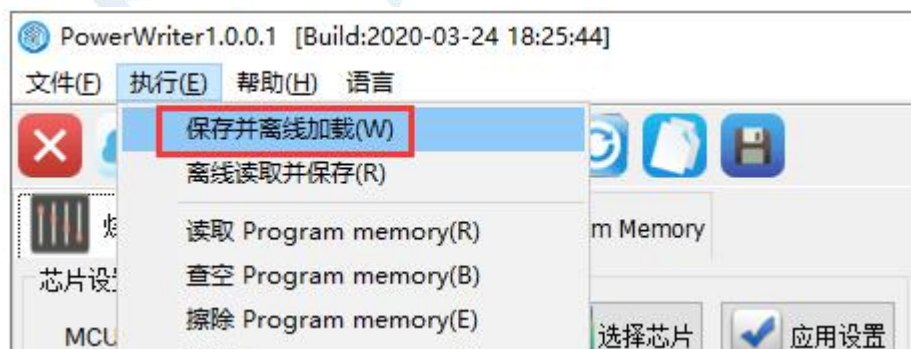
- ①重新编译；
- ②源代码中的矩阵和 ICWKEY 中的设定是完全一样的；
- ③成功生成新固件(烧录档案)

1.5 PowerWriter 保存项目



①添加刚生成的固件;

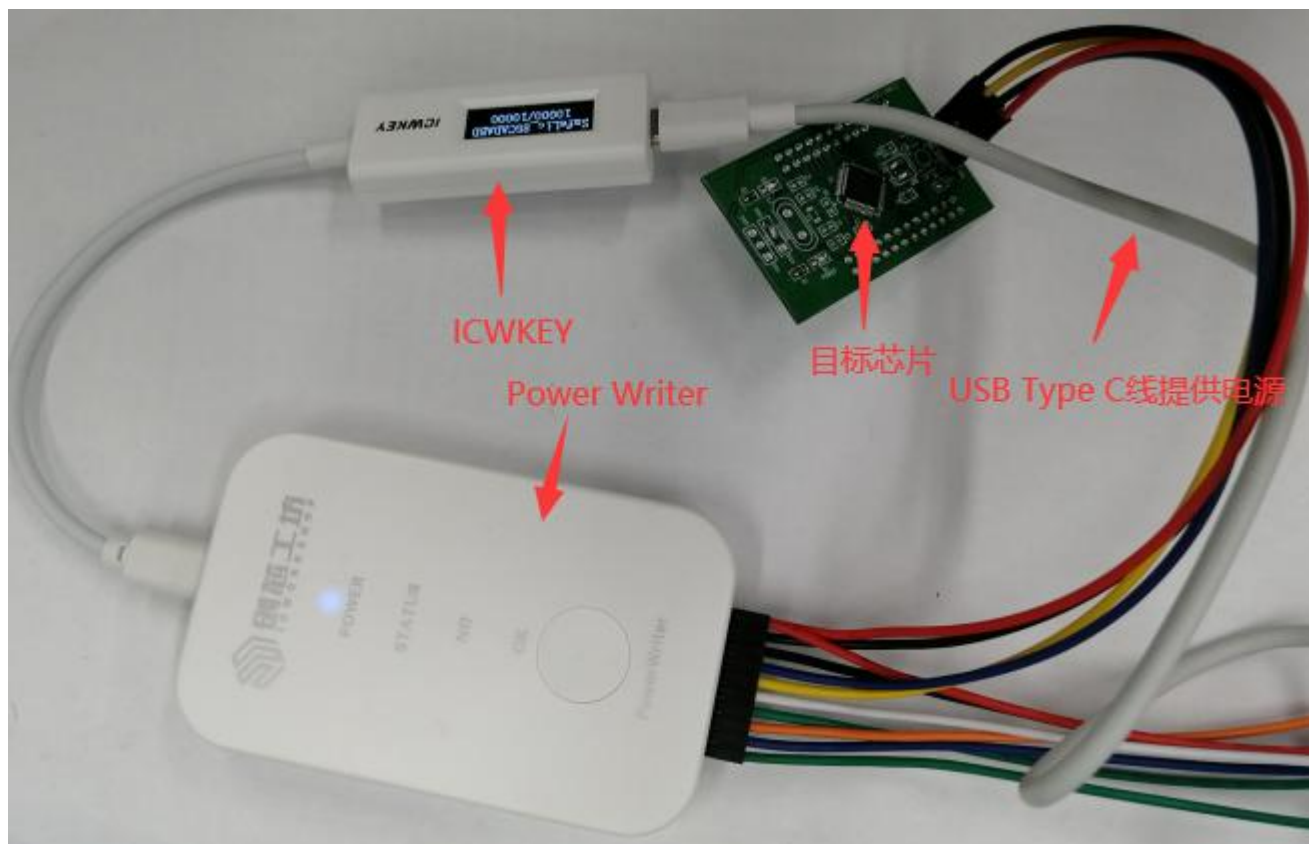
②应用固件。



将项目设置和固件下载到 PowerWriter 烧录器。

1.6 离线烧录目标芯片

每执行一次烧录，ICWKEY 会将剩余授权次数减一，OLED 屏幕会同步更新显示



1.7 验证烧录

范例程序功能为 LED 闪烁，如果将目标芯片程序读出直接烧录到另外一个芯片，LED 不会闪烁，UID 保护生效。

2 安全机制

2.1 ICWKEY 自身的安全性

ICWKEY 使用的安全机制：

1, ICWKEY 主控芯片采用 ST 高阶芯片开发，开启了 2 级读保护，Debug 功能 (JTAG & serial wire) 已经永久禁用，芯片内的数据无法被外部调试器读出；

- 2, ICWKEY 与 PC/PowerWriter 的通讯采用 AES 加密, 内置防暴力破解策略, 从外部无法枚举出密钥;
- 3, ICWKEY 内置了 UID 高阶加密算法, 即使程序被暴力破解被盗, 也无法在其他同型号芯片上运行;
- 4, PowerWriter 同样设置了授权次数, 单独破解 ICWKEY 并不能去除授权次数限制;
- 5, 授权次数信息有备份区, 授权过程中如果掉电, 剩余授权次数不会丢失或重置;
- 6, 授权次数信息的存储地址有做寿命管理, 不用担心因为授权次数的频繁更新, 导致内部 flash 擦写次数寿命耗尽。
- 7, ICWKEY.exe 采用 c++ 11 开发, 内部代码采用了大量的检测技术, Hash/加密关键数据, 用于检测内存中的软件数据是否被读取, 篡改等异常操作, 采用商业保护软件加密, 做了大量的逆向分析检测, 多层防护。

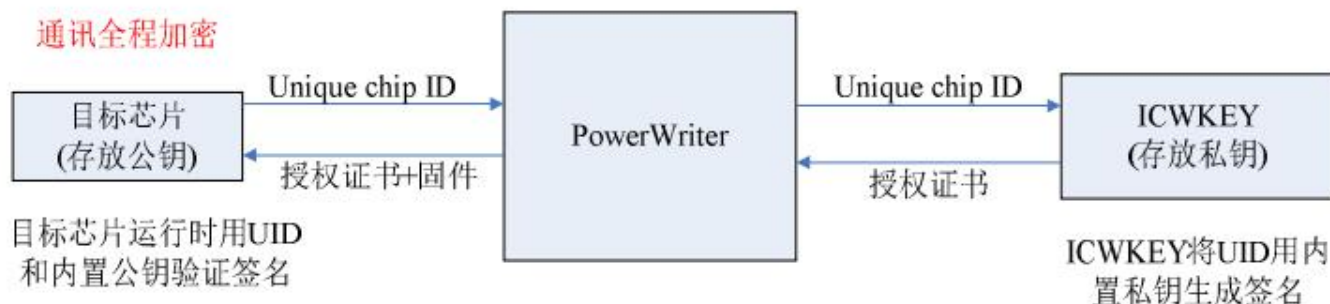
2.2 UID 向量矩阵加密(Matrix)介绍

这是一种常见的加密算法, 基于目标芯片内的 unique chip ID, 与用户自定义的 Key 进行加减乘除, 与或非异或等运算后, 计算出一组授权密钥, 存储到芯片内, 芯片运行时使用同样的计算方式验证授权密钥是否正确, 运算符负荷和资源开销非常小, 效率非常高。市面上常见的烧录器 UID 矩阵加密, 只能提供少数几组固定矩阵, ICWKEY.exe 可以随机生成矩阵, 用户也可以手动修改矩阵, 完全无法被枚举出来, 下图是一组示例:



2.3 UID 椭圆曲线数字签名(ECDSA)介绍

椭圆曲线加密算法(Elliptic Curve Cryptography)，简称 ECDSA，是基于椭圆曲线数学理论实现的一种非对称加密算法。相比 RSA，ECDSA 优势是可以使用更短的密钥，来实现与 RSA 相当或更高的安全。安全性远比向量矩阵加密高，但是运算负荷和资源消耗大，内置算法采用纯软件方式，目标芯片验证签名时间较长，芯片 64MHz 主频验签耗时约 0.6s，芯片 SRAM 堆栈需要设置到 12Kbytes(代码有使用动态内存分配)，这是一种更高的代价获得更高安全性的选择。



ECDSA证书生成器

```

//Public key
const static uint8_t PUBLIC_KEY[49]={
    0x04,0x02,0xED,0x53,0x47,0x72,0x9A,0x74,0xC6,0x7C,0x7A,0xFC,0x1C,0x02,0x82,0xA1,
    0x05,0xB4,0x36,0xC9,0x49,0x59,0xD8,0x45,0xBB,0xAC,0x06,0xF5,0x5B,0x6A,0x38,0xBB,
    0xD0,0x00,0xEF,0x4F,0x6D,0xB6,0x57,0xDF,0x28,0x95,0x54,0xC3,0xD3,0xF3,0x35,0x10,
    0xD3
};

//Private key
const static uint8_t PRIVATE_KEY[24]={
    0x30,0x72,0x03,0x59,0xD8,0xBD,0x4A,0x73,0x7F,0x0E,0xFA,0x99,0x0E,0x1F,0xF0,0xBA,
    0x41,0xB7,0xED,0xF9,0x90,0x32,0xE7,0xDA
};
        
```

存放目标芯片

存放ICWKEY

使用说明

公钥保存到目标芯片，私钥保存到授权安全盾，烧录时授权安全盾将目标芯片的chip ID用私钥生成签名证书，烧录到目标芯片；使用时目标芯片利用自己的chip和公钥验证签名

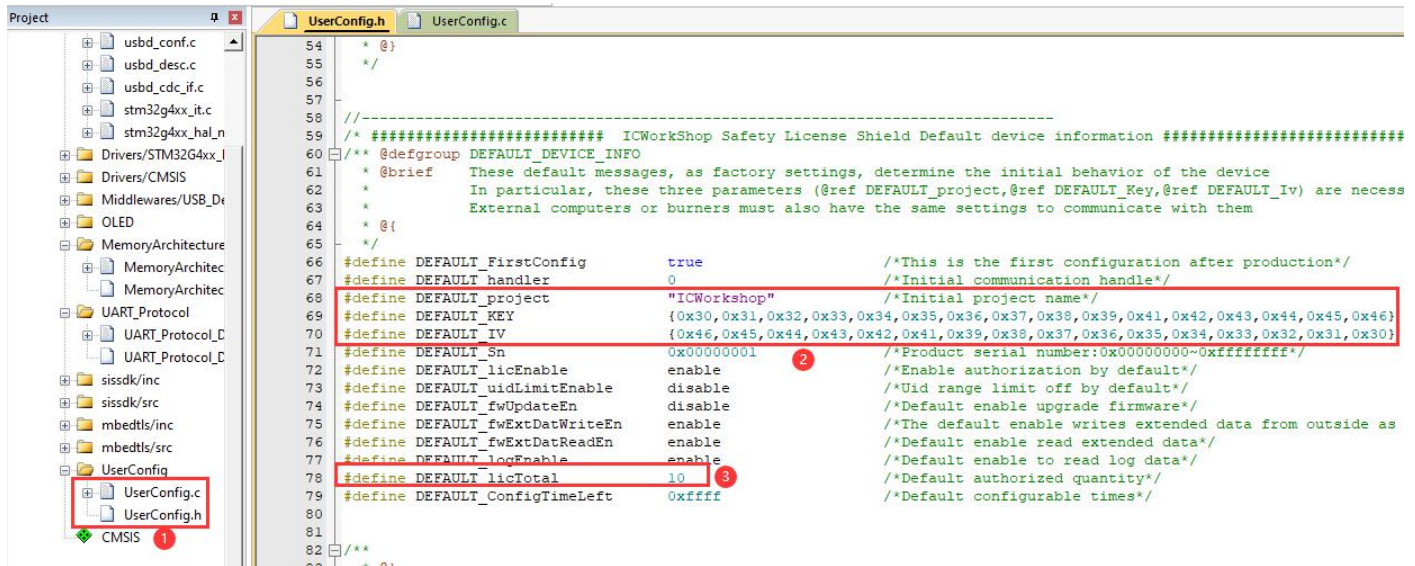
随机生成

导出源码

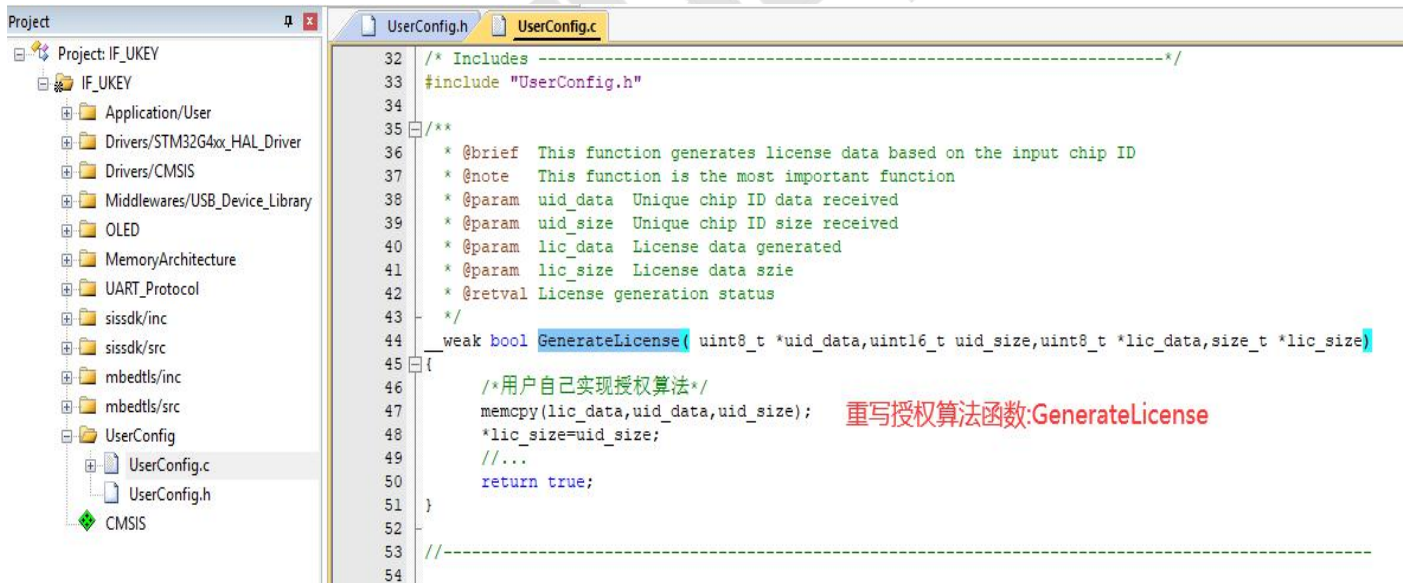
编译并保存

2.4 如何开发自定义加密算法

如果开发者希望使用 ICWKEY 开发自定义加密算法，请联系我们（cs@icworkshop.com）。我们可提供 SDK，SDK 实现了必要的通讯协议，开发者只需要设置自己的项目名称和通讯密钥，重写授权函数即可。



- ①用户配置文件;
- ②设置项目名称和通讯密钥;
- ③设置授权次数



2.5 安全优化的 UID 授权使用范例

通常情况下，要破解固件，目标芯片的 chip ID 地址和 UID 授权的存放地址是破解入口，可以用来定位 UID 校验函数的位置，我们提供的 UID 授权使用范例程序进行安全性能的优化，编译后的固件中不会明文出现这两个关键信息。

3 软件功能详细介绍

软件界面截图



3.1 菜单

1. 文件

新建项目/加载项目/保存项目/项目另存为，项目内容包含设备配置和 UID 算法，项目文件加密存储

退出：关闭软件



打开的项目名和路径信息将显示在标题栏，如果有未保存的修改，将出现“*”提示。

2. 执行

默认通讯设定：使用默认的通讯设定，其设定和 ICWKEY 出厂设置一致，项目名称=" ICWorkshop"，

新密码=30313233343536373839414243444546，新向量=46454443424139383736353433323130；

启用项目通讯设定：通讯设定采用项目配置窗口中的新项目名称&新密码&新向量；

项目保存并更新：弹出项目保存对话框，保存完项目文件后，将项目内容下载到 ICWKEY；

3. 帮助

官方网站/许可协议/用户手册，用户可以从中获得帮助信息；

4. 语言

简体中文/英语，实现中英文切换；

3.2 通讯设定

软件和 ICWKEY 建立通讯必需的通讯参数，点击“连接设备”按钮，实现和 ICWKEY 的连接/断开，请留意 ICWKEY OLED 屏幕上面显示的项目名称是否一致。

3.3 项目配置->设备配置

新项目名称&新密码&新向量：应从 PowerWriter.exe 复制过来；

设备序列号： ICWKEY 的产品序列号，每完成一次 ICWKEY 更新，序列号自动加一，用户可手动更改；

UID 最小值&UID 最大值： 当使能“限制 UID 授权范围”时候有效，用户可以通过此功能设定 ICWKEY 只能对此 UID 范围内的芯片进行授权。

授权数量： ICWKEY 最多能够进行多少次烧录授权；

可配置次数： 用户可以决定 ICWKEY 的可配置次数，新的设定必须比实际剩余配置次数小才有效；

实际剩余配置次数： 只读项，从连接着的 ICWKEY 中获取；

使能授权工具： 开/关授权功能；

允许固件升级： 开/关固件升级，如果有更新的固件，当软件连接到 ICWKEY 后会提示用户是否需要升级；

限制 UID 授权范围： 开/关限制 UID 授权范围功能；

允许写入 UID 算法： 开/关写入 UID 算法功能；

开启日志记录： 开/关日志记录，日志记录信息只包括授权总数，已用次数，成功次数，失败次数，未知错误；

读取目标配置： 执行一次读取 ICWKEY 内的项目配置

3.4 项目配置->UID 算法



用户可以选择一种 UID 加密算法，并进行详细配置。

3.5 项目配置->日志信息

项目配置

设备配置 UID算法 日志信息

日志信息

授权总数 10000 已用次数 11

成功次数 11 失败次数 0

未知错误 0

LOG 读取日志

测试授权

Chip ID(L->H) 0x 352DDA054348393242510543 TEST 测试授权

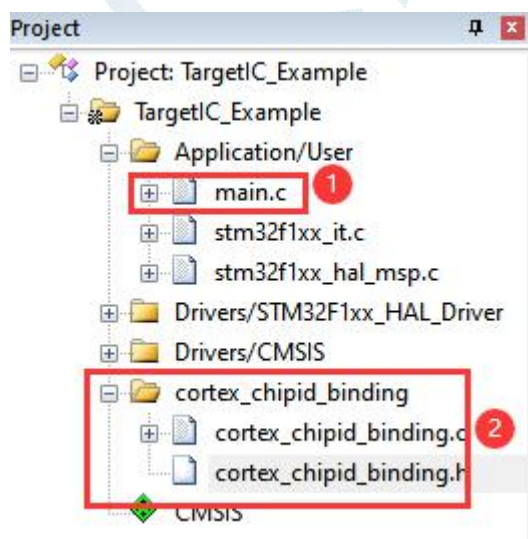
提供读取日志和测试授权功能。

4 UID 授权目标芯片使用范例

范例程序基于 STM32CubeMX 开发，方便开发者二次移植。

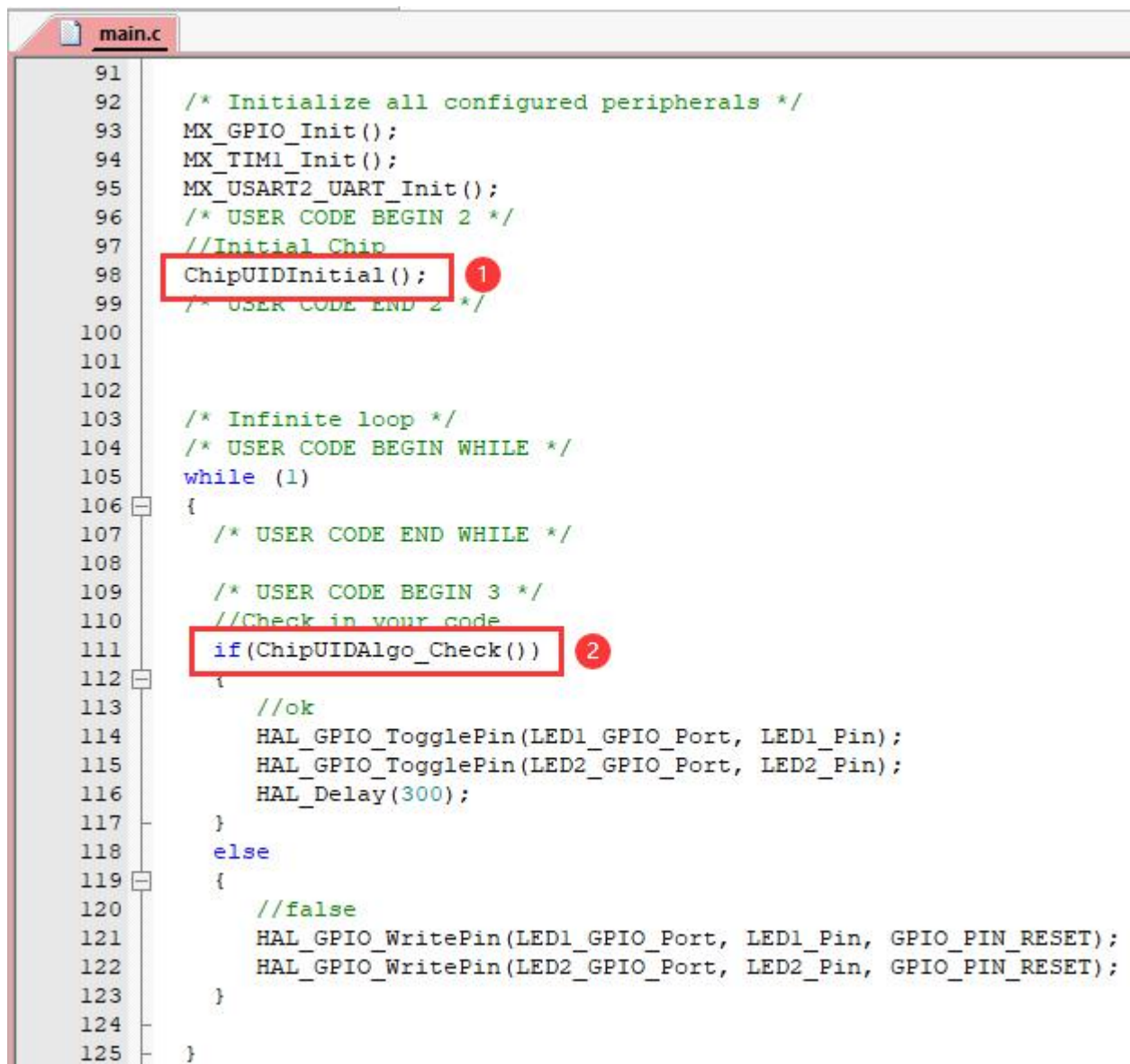
4.1 向量矩阵加密 (Matrix) 目标芯片程序范例

范例路径: TargetIC_Example\TargetIC_Example_Matrix_v1.03



①在 main.c 中演示了如何进行初始化和调用 UID 校验。

②cortex_chipid_binding.c&cortex_chipid_binding.h 是由 ICWKEY.exe 自动生成，实现授权密钥的验证；



```
91
92  /* Initialize all configured peripherals */
93  MX_GPIO_Init();
94  MX_TIM1_Init();
95  MX_USART2_UART_Init();
96  /* USER CODE BEGIN 2 */
97  //Initial Chip
98  ChipUIDInitial(); 1
99  /* USER CODE END 2 */
100
101
102
103  /* Infinite loop */
104  /* USER CODE BEGIN WHILE */
105  while (1)
106  {
107      /* USER CODE END WHILE */
108
109      /* USER CODE BEGIN 3 */
110      //Check in your code
111      if (ChipUIDAlgo_Check()) 2
112      {
113          //ok
114          HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
115          HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
116          HAL_Delay(300);
117      }
118      else
119      {
120          //false
121          HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
122          HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
123      }
124
125  }
```

①在进行 UID 校验之前必须先调用 ChipUIDInitial 函数，该函数用于获取当前芯片的 unique chip ID；

②校验 UID 授权，此函数是内联的，用户可以在程序多个位置进行调用。

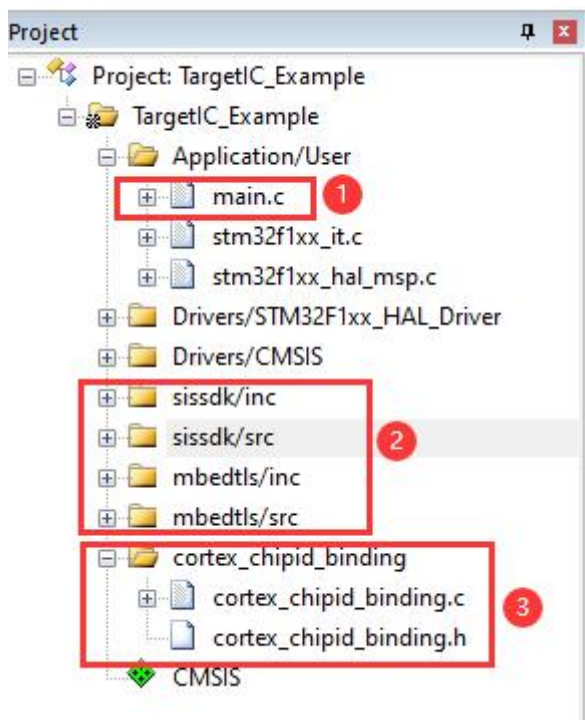
注意事项：

1，因为 ICWKEY 只负责授权，并不理会是什么目标芯片，cortex_chipid_binding.h 中芯片 chip ID 的存储地址和 UID 授权的存储地址，必须和实际一致，需要用户手动修改；

2，需要特别注意，STM32 有部分型号 chip ID 的存储地址不是连续的，用户需要额外进行处理。

4.2 椭圆曲线数字签名 (ECDSA) 目标芯片程序范例

范例路径：Examples_for_mdk\TargetIC_Example_ECDSA_v1.03



①在 main.c 中演示了如何进行初始化和调用 UID 校验;

②ECDSA 需要的库文件, 也提供了 CRC, ARS 等其他功能;

③cortex_chipid_binding.c&cortex_chipid_binding.h 是由 ICWKEY.exe 自动生成, 实现授权密钥的验证;

除了向量矩阵加密的注意事项, 还需要额外注意:

1) 将栈设置为 $\geq 4\text{KB}$, 堆设置为 $\geq 8\text{KB}$, 这是一个安全的数值, ECDSA 对 RAM 堆栈的需求较大;

2) Keil 环境下必须参考范例, 在 Option \rightarrow C/C++ \rightarrow Include Paths 导入必需的附加路径;

3) ECDSA 包含文件很多, 不用刻意去删除掉一些文件, 虽然都会被编译, 但链接时未使用的文件不会被链接进去, 这些文件中还包含 CRC 校验, AES 加解密库, 给开发者提供更多的帮助。

5 注意事项

5.1 导出的 UID 授权算法源代码必须根据目标芯片进行修改

ICWKEY.exe 导出的源代码必须根据目标芯片实际情况进行修改, 在 cortex_chipid_binding.h 中修改 3 个地方:

```
#define    UID_CHIP_MASK                0x1155282C                //Random generation

#define    UID_CHIP_SIZE                12                        //ChipID Size
```

```
#define    UID_CHIP_ADDR                (0x1FFFF7E8^UID_CHIP_MASK) //ChipID Inner Addr in chip

#define    UID_KEYADDR_INNER            (0x0800FF80^UID_CHIP_MASK) //Key Store Addr In flash
```

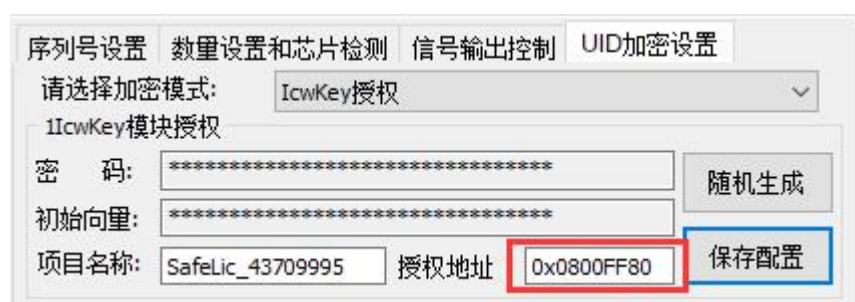
解释:

UID_CHIP_SIZE=Chip ID 的长度，通常都是 12bytes;

UID_CHIP_ADDR=Chip ID 在芯片中的存储地址，需要通过查看芯片厂家的数据手册获得;

UID_KEYADDR_INNER=加密算法生成的授权密钥存放在芯片中的物理地址，烧录器负责将授权密钥烧录到此地址，这个地址由用户自己决定。

UID_CHIP_MASK=ICWKEY.exe 每次导出源码时生成的随机数，起到混淆的作用，固件中不会明文出现芯片的 chip ID 地址和 UID 授权地址，增加了被破解的难度。



Chip ID 在芯片中的存储地址，请参考“Power Writer 用户参考手册 RM0001.pdf”的“Power Writer UID 地址对照表格”，下表仅列出部分型号供用户参考，实际开发时应以芯片厂家官方手册为准。

芯片类型	芯片系列	UID 长度(字节)	UID 起始地址
STM32	STM32F0XX	12	0x1FFFF7AC
	STM32F1XX	12	0x1FFFF7E8
	STM32F2XX	12	0x1FFF7A10
	STM32F3XX	12	0x1FFFF7AC
	STM32F4XX	12	0x1FFF7A10
	STM32F7XX	12	0x1FF07A10 (F72XX/F73XX) 0x1FF0F420 (F74XX/F75XX/F76XX/F77XX)
	STM32G0XX	12	0x1FFF7590
	STM32G4XX	12	0x1FFF7590
	STM32H7XX	12	0x1FF1E800
	STM32L0XX	12	0x1FF80050 Offset 0x00:UID[31:00]

			Offset 0x04:UID[63:32] Offset 0x14:UID[95:64]
	STM32L1XX	12	0x1FF80050 (Cat. 1&Cat. 2) 0x1FF800D0 (Cat. 3&Cat. 4&Cat. 5&Cat. 6) Offset 0x00:UID[31:00] Offset 0x04:UID[63:32] Offset 0x14:UID[95:64]
	STM32L4XX	12	0x1FFF7590
	STM32L5XX	12	0x0BFA0590
	STM32WB55XX	12	0x1FFF7590

```

32  ----- */
33
34  /* Includes ----- */
35  #include "cortex_chipid_binding.h"
36
37  #warning Please confirm whether the UID_CHIP_ADDR&UID_KEYADDR_INNER is correct. We
38

```

cortex_chipid_binding.c 中有放置一条警告，用来提醒用户必须确认芯片 chip ID 地址和授权地址是否设置正确，如果用户觉得多余，可以屏蔽它。

5.2 务必保管好项目文件和密码

电脑通过 ICWKEY.exe 加载项目文件 (*.uprj) 才能连接和配置 ICWKEY，打开项目文件需要密码，所以务必保管好项目文件和密码，否则 ICWKEY 无法再次配置。

5.3 UID 授权在芯片中的存储位置不能和用户代码区重叠

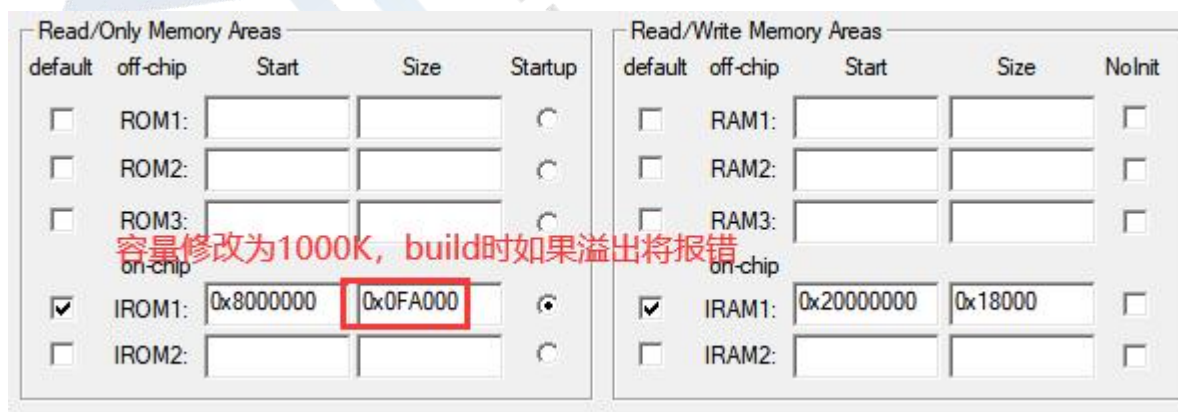
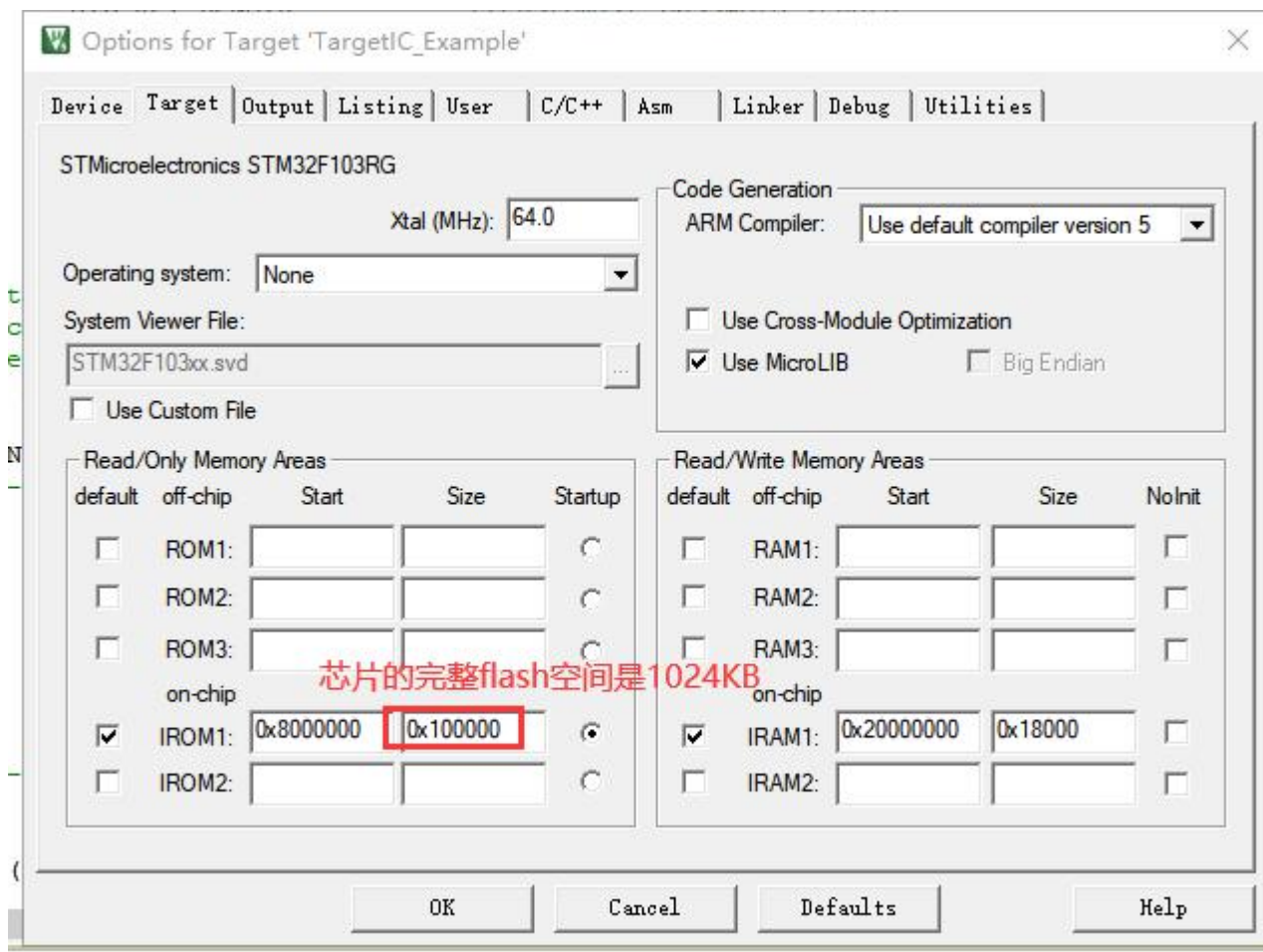
PowerWriter 烧录时会把 UID 授权密钥烧录到设定的授权地址，如果此地址区间已有用户的代码或数据，将被覆盖，可能导致程序运行异常，用户可以采用两种方式避开此问题：

1，将授权地址设定在芯片存储空间末尾位置，并且确保用户的代码或数据不会占用到此地址空间，

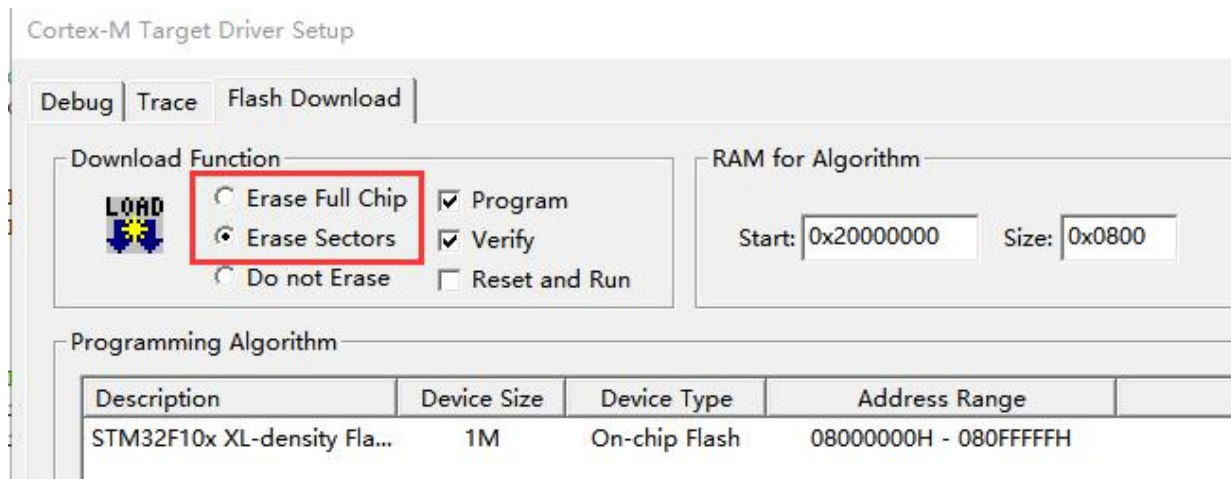
举例：芯片有 1024KB 空间，用户程序空间只有 1000KB，末尾的 24KB 是未使用空间的，可以将授权地址设置

在此空间。

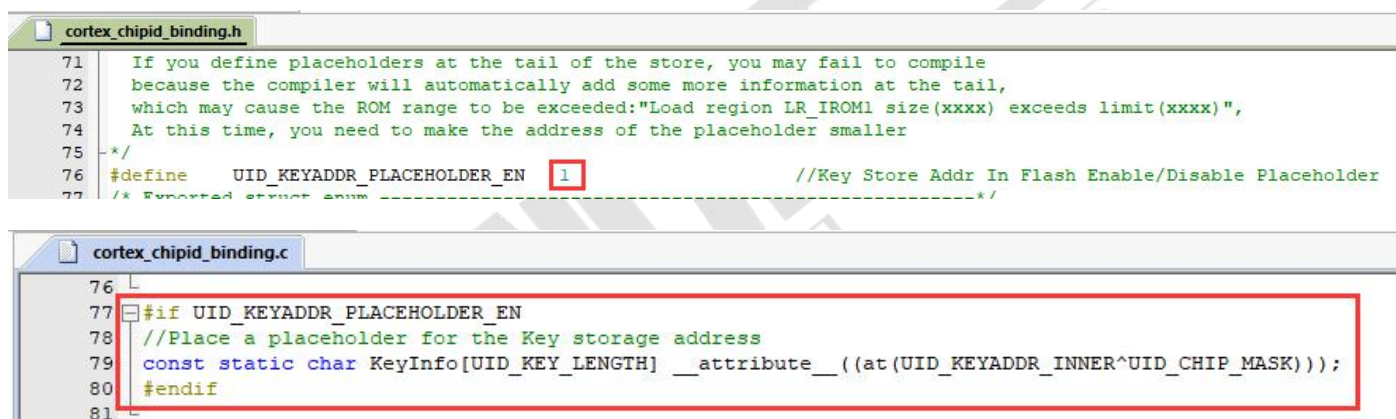
缺点：用户必须清楚自己的程序大小，确保授权地址是在未使用空间，比较合理的做法参考如下：



优点：如果芯片通过 PowerWriter 烧录过一次，则 UID 密钥/证书将一直保存，用户调式程序时候不会将其覆盖或者擦除掉，保证调试时候 UID 校验始终是通过的，注意调试器的设定，应选择“块擦除”，不能选择“整片擦除”！



2, 在程序中预留存放 UID 授权的数组空间, 只需在代码中将 UID_KEYADDR_PLACEHOLDER_EN 的定义由 0 改成 1 即可实现;



如果用户使用此功能, 则建议将 UID 授权的存放地址设置为 flash 地址空间中靠前的位置;

缺点: 数组内容为空, 调试时 UID 校验会失败, 有两种解决方法:

- 1) 调试时屏蔽 UID 校验, 最终发布 release 版本时再开启 UID 校验;
- 2) 预先知道授权内容, 填入到数组, 调试时可以确保当前目标芯片校验通过, release 时不用再额外处理。可以先用 PowerWriter+ICWKEY 烧录一次芯片, 再回读数据, 找到授权地址, 获得当前芯片的授权密钥。或者使用 ICWKEY.exe 的测试授权功能, 能够快速知道授权密钥的具体数据。



优点: 不用担心烧录器烧录 UID 授权数据会破坏程序或数据。

6 常见问题

6.1 USB 驱动安装不成功

如果电脑操作系统是 WinXP, Win7 或 Win8, 且非官方原版, 而是 ghost 镜像或者精简系统, 可能会遇到安装失败的问题, 可以在网络上搜索 “ST 虚拟串口驱动安装失败” 获取解决方式, 下面是一个解决案例:

<https://blog.csdn.net/oshan2012/article/details/84063946>

不要选择与实际系统的不相符的驱动进行安装。

6.2 如何判断 PowerWriter 和 ICWKEY 连接成功

确保 ICWKEY.exe 和 ICWKEY 断开连接(未配对)的情况下, ICWKEY 插入 PowerWriter 的 USB 插座, 蜂鸣器滴滴两声就表示连接成功。

6.3 如何确认数据已经正确烧录到目标芯片

当 PowerWriter 烧录成功后, 绿色指示灯会亮, 如果用户担心烧录的数据与期望的不一样, 在芯片没有开启读保护的情况下, 可以使用 PowerWriter.exe 或者其他工具对目标芯片进行校验, 或者回读数据观察数据是否和原始档案一致

6.4 为什么加入了 UID 校验, 程序调式时不能通过校验

UID 授权密钥是 PowerWriter 烧录进去的, 调试时还没有授权数据, 所以校验不能通过, 解决方法参考 5.3 章节

6.5 程序应该预留多大空间存放 UID 授权密钥

向量矩阵算法, 授权数据不会超过 chip ID 的长度, 通常是 12bytes;

椭圆曲线数字签名, 授权数据不超过 141bytes (当前版本是 141bytes, 不排除后续版本有更新)。

6.6 烧录失败的原因

1) 接线问题, 接线错误, 接线松动;

- 2) 配置出错，选择的芯片和目标芯片不一致；
- 3) PowerWriter 或 ICWKEY 设置的授权次数已耗尽；
- 4) 目标芯片已经关闭了烧录功能，比如 STM32 开启了二级读保护，JTAG & serial wire 已经被禁用；
- 5) 目标芯片之前已经有烧录过程序，其程序上电立即将把烧录口烧录功能关闭，改成 GPIO，这种情况需要将目标芯片的复位脚引出来，让芯片在复位状态再执行烧录；
- 6) 烧录器供电不足，电压过低可能会导致烧录失败，比如老式台式机，机箱前面的 USB 口驱动电流会比后面的小。

6.7 烧录后二次回读或校验失败

可能是目标芯片已经开启了读保护。

7 联系我们

创芯工坊科技（深圳）有限公司

网址: www.icworkshop.com

电话: 400-1568-598

邮箱: cs@icworkshop.com

官网二维码

微信公众号



8 版本历史

版本	日期	说明
V1.00	2020/3/6	首次发布