版本：**Next**

2：概要信息

2.1 功能参数

ICWKey 功能参数

2.2 软件安装

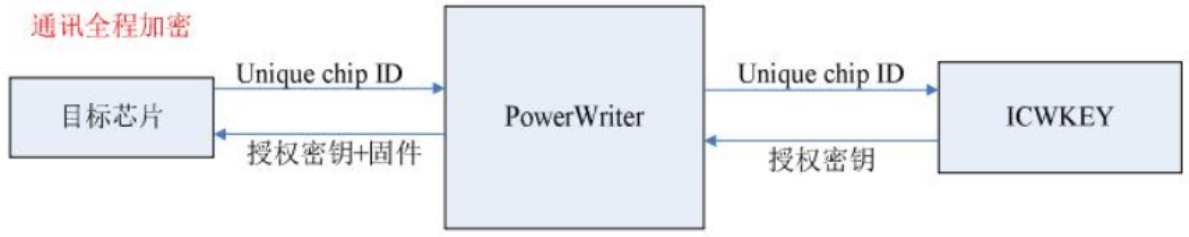
ICWKey 软件

版本：Next

2.1 功能参数

2.1.1 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 提供向量矩阵加密(Matrix)、ECDSA 数字签名两种 UID(Unique Chip ID)授权算法、SDK 供用户开发自定义授权算法，以满足不同的需求。 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

> **提示**
>
> **MCU 通用高级软件保护库**” 可提供更高级别的防护、集成ICWKEY 签名、固件加壳、

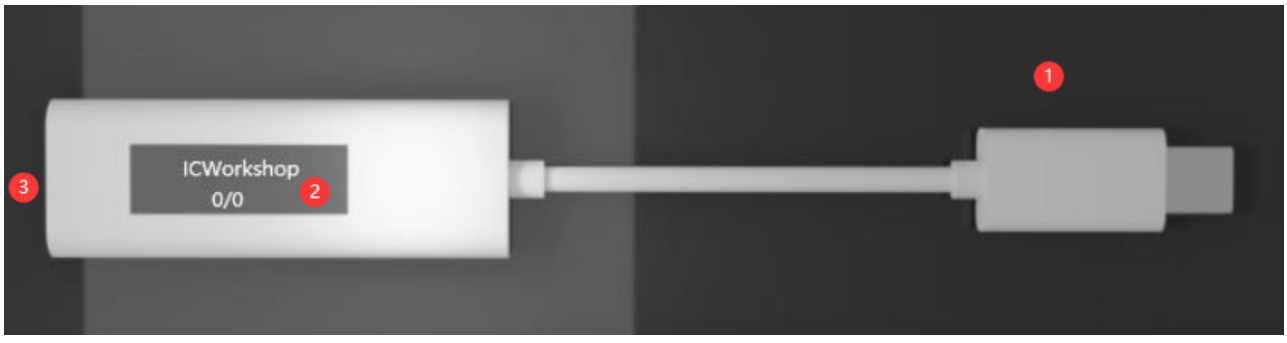
> 固件压缩、函数级代码加密、固件验证、对象监控、非法访问检测、调试器检测、权限分离与控制等丰富的安全特性、可进一步提升软件的安全、详情请发送邮件到 cs@icworkshop.com 获取更详细的资料（**为防止被滥用、当前未公开发布**）。

2.1.2 产品参数

- **产品尺寸**：57mm x 22.5mm x 10.6mm (≈)
- **工作电压**：5V (**USB Type-C**)
- **产品功耗**：60mA~ 90mA

> **提示**
>
> 产品的参数为理论数据，因为批次、工作环境、产品改进等原因，实际可能存在差异，仅供参考，如有更改，恕不另行通知！

2.1.3 接口信号



- ①：PowerWriter® Type-C 主机端口(**连接到PowerWriter**)。
- ②：ICWKEY OLED 显示屏
    - 项目名称：显示格式为：**SafeLic_xxxxxxxx**，xxxxxxxx 为随机项目名hash。

    - 剩余次数/总次数：如998/1000，表示可用签名次数剩余 998 次，总签名次数为1000。

- ③：Type-C 从机端口：ICWKEY 供电通讯（**连接到PC**）。

2.1.4 功能特性

- [x] 可限制Unique ID 签名范围
- [x] 可控制签名数量
- [x] 可配置次数（重复使用控制）
- [x] 授权日志查询
- [x] 签名测试
- [x] 多语言
- [x] 签名支持
    - [x] Matrix 签名
    - [x] ECDSA 签名

2.1.5 安全特性

- ICWKEY 采用安全芯片开发，**并集成高级软件保护库**，保护固件安全。
- ICWKEY 与 PC / ICWKEY 与 PowerWriter 通讯加密，内置防暴力破解机制，无法通过穷举破解密码。
- 数据双区、加密设计，掉电应急存储。
- 超长寿命设计。

版本：Next

2.2 软件安装

2.2.1 简介

ICWKEY 客户端提供ICWKEY设备的完整访问、配置接口。

2.2.2 客户端安装

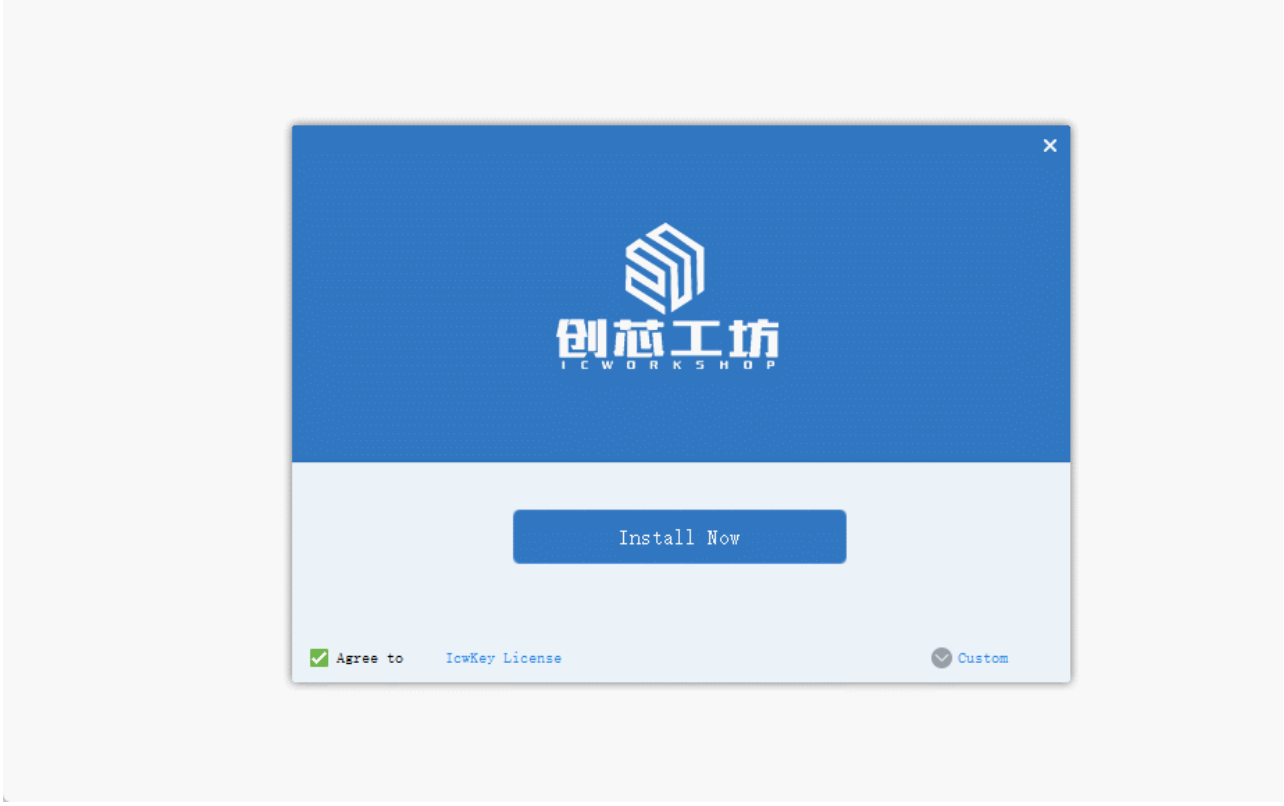
2.2.2.1 软件下载地址

最新客户端下载地址见官网站点：

https://www.powerwriter.com/index/index/products?p=21&c=files&t=Client

请根据当前使用的系统平台下载ICWKey 的安装包.

2.2.2.2 软件安装流程



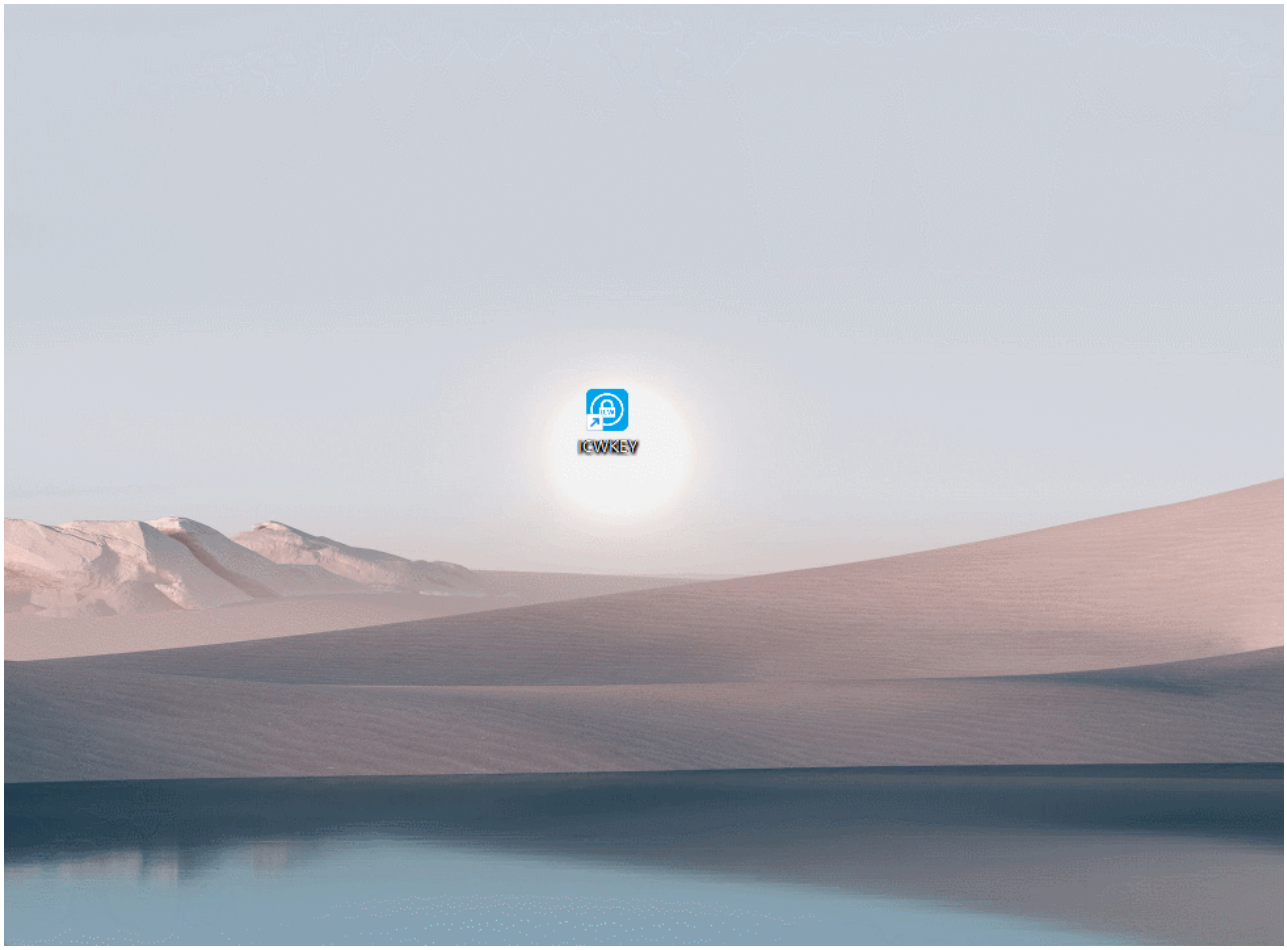
2.2.2.3 快捷启动

- 从系统桌面找到 ICWKEY 图标启动。
- 从快捷搜索栏搜索ICWKEY， 然后进行启动。

2.2.3 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， 提示需要安装驱动， 如果电脑是 Windows 10系统， 系统会自动完成驱动安装， 如果是比 Windows 10 更早的系统可能需要手动安装， 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 启动安装，演示如下所示



编辑本页

版本：Next

3：快速开始

3.1 PowerWriter配置

ICWKey 基础配置

3.2 ICWKEY配置

ICWKey 配置

3.3 ECDSA示例

详细演示ECDSA签名算法的使用，以及注意事项

3.4 Matrix示例

详细演示Matrix签名算法的使用，以及注意事项

版本：Next

3.1 PowerWriter配置

3.1.1 描述

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， 此外PowerWriter端需要配置签名的**写入地址**。

3.1.2 PowerWriter配置

配置流程参考如下：

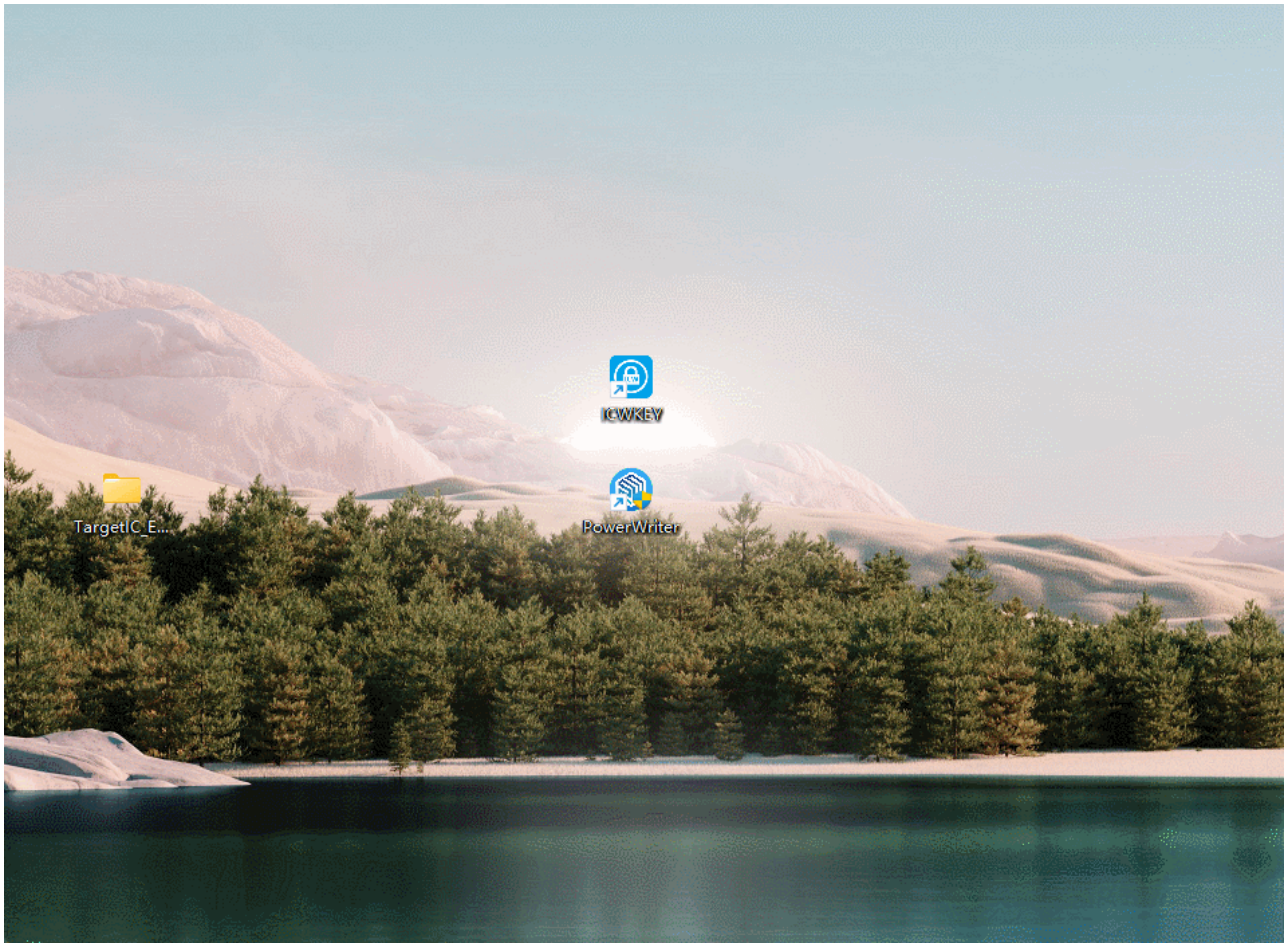
- 打开PowerWriter软件， 加载已有项目或者选择需要签名的芯片新建项目。
- 选择烧录器设置页面-> 授权与签名-> 请选择签名模式栏中选择：**ICWKEY 授权（或锁定模式）**。
- 修改**授权地址**：修改授权地址为实际存储签名信息的地址（存储在**固件中的位置**， 如设置为0x08002000 表示签名信息需要存储在 0x08002000的位置）。
- 设置完成后， 保存PowerWriter 项目， 避免信息丢失。

> **警告**
>
> - 授权地址为实际存储签名信息的地址， 每一个项目均不相同， PowerWriter 首次选择芯片时， 会将地址设置为固件的尾部， 请根据实际情况进行调整。
> - 设置完成后， 请保存PowerWriter 项目， 以免配置信息丢失， 无法连接ICWKEY设备。
> - **锁定模式**补充说明：锁定模式在下次重新加载项目之后， 无法再次查看和修改通

> 讯配置（**签名地址可修改**）。

3.1.3 PowerWriter配置演示

PowerWriter 端配置流程演示如下所示。



编辑本页

版本：Next

3.2 ICWKEY配置

3.2.1 ICWKEY配置

配置流程参考如下：

- ☑ 连接ICWKEY设备（新设备用默认的通讯配置、重复使用的设备加载之前的项目文件）
- ☑ 复制PowerWriter 配置端的通讯密码和项目名称到ICWKEY 的配置端。
- ☑ 设置授权次数（可控制实际可执行多少次签名）
- ☑ 检查并配置可配置次数（**如希望设备重复使用、可不用调整、如使用一次后作废，请配置成一次**）
- ☑ 在UID 算法页面选择签名算法，导出算法源码，并保存。
- ☑ 点击保存并更新按钮，对ICWKEY 进行配置。
- ☑ 根据弹出窗口，保存ICWKEY 项目文件。

> **警告**
>
> - 通讯配置由PowerWriter生成，复制到ICWKEY的配置窗口。
> - UID 算法选择，请先生成->导出源码->保存设置，后续需要基于导出源码做开发集成。
> - ICWKEY 配置好之后，请保存好项目，并记住项目密码，丢失项目（连接信息），将无法连接到ICWKEY设备。
> - 可配置次数特别说明：默认为65535次可重复使用次数，每更新一次设备，则次数-1，当次数达到0 时，此时，ICWKEY 设备，将不能更改任何信息。

3.2.2 ICWKEY配置演示

ICWKEY 端配置流程演示如下所示。

编辑本页

版本：Next

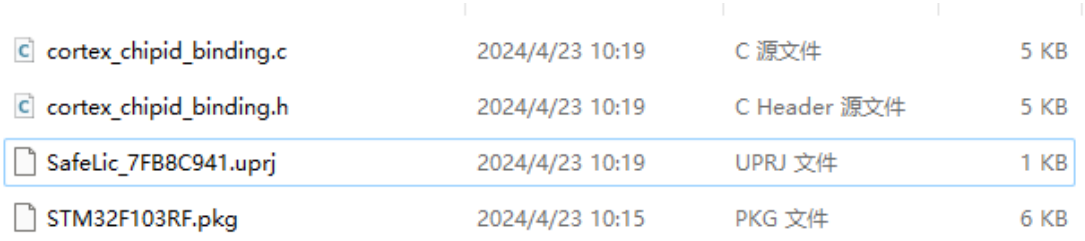
3.3 ECDSA示例

3.3.1 准备

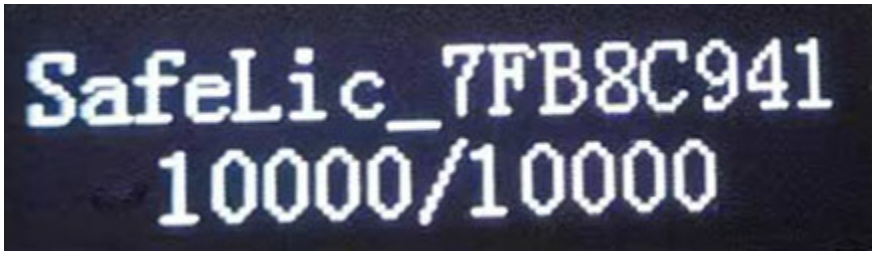
ECDSA 签名为一种非对称加密的电子签名方法，私钥存储到签名设备ICWKEY 中，公钥存储在项目固件中，ICWKEY 通过对目标芯片ID 以及当前的私钥，生成签名信息，然后通过PowerWriter 将签名信息写入到固件指定地址，固件运行时，通过公钥 + ID去验证当前签名信息是否有效，从而判定当前芯片是否已经得到有效的授权，避免固件被直接拷贝使用，在开始之前，我们需要按照流程核查所有的准备工作都已经完成。

- ☑ PowerWriter 项目中使用ICWKEY 签名(或者 ICWKEY 签名锁定模式)。
- ☑ 已经设置签名地址（如0x08002000）。
- ☑ PowerWriter 端的通讯信息已经同步到ICWKEY 的项目中，并与项目重新建立加密通讯。
- ☑ 合理设置可授权次数，比如设置为10000。
- ☑ 签名方法：选用ECDSA 签名，保存到ICWKEY，并导出了源码。

如以上步骤均完成，则可以看到ICWKEY 设备的显示信息，以及导出的源码信息，参考如下：



同时ICWKEY 的设备将显示如下的信息：



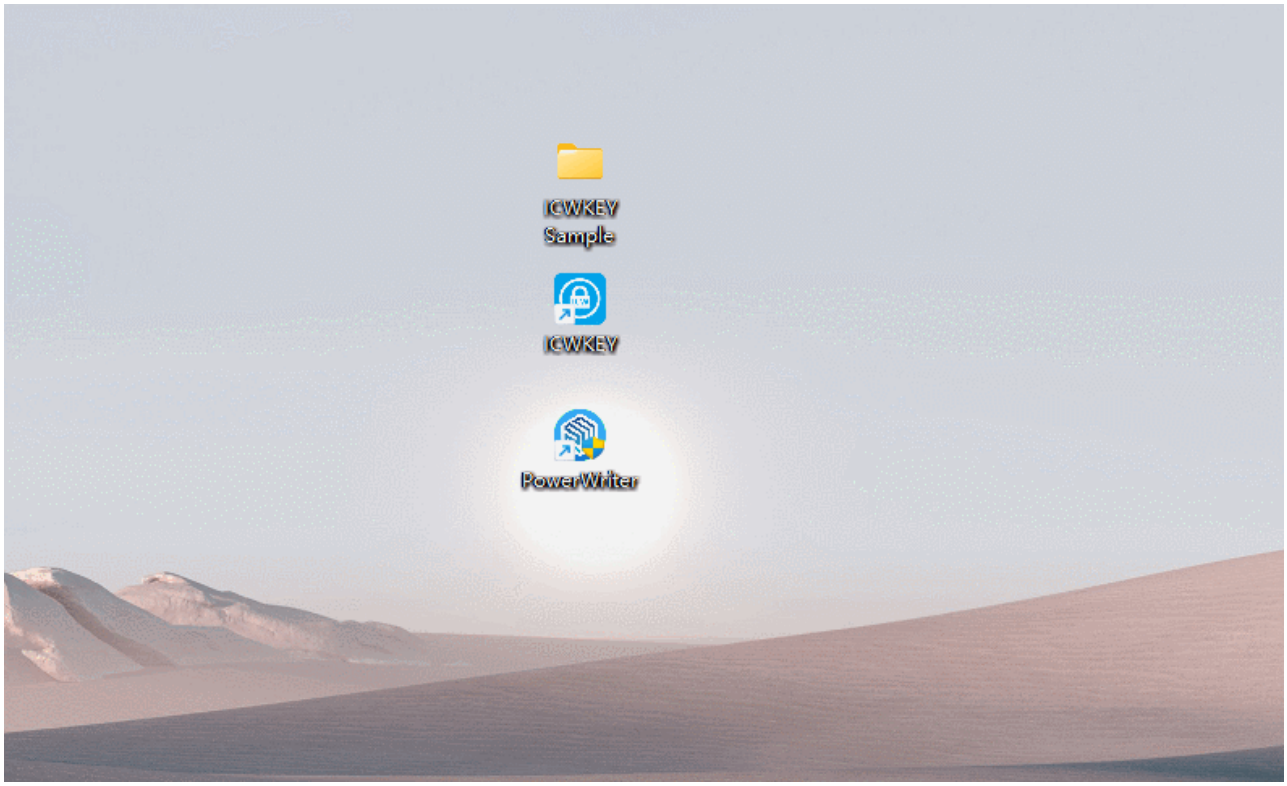
3.3.2 示例工程

3.3.2.1 准备

示例工程路径ICWKEY 的安装路径下，具体为：

```
C:\Users\用户名\AppData\Local\ICWKEY\Examples_for_mdk
```

可通过ICWKEY 桌面图标，快速定位到为止，并拷贝ECDSA 示例工程到指定路径，并解压，参考演示如下：



3.3.2.2 代码结构

3.3.2.2.1 startup_stm32f103xg.s

- [x] 调整堆大小 > 0x1300
- [x] 挑战栈大小 > 0x800

```
Stack_Size      EQU     0x1000      ;请将堆栈调大一些，ECDSA 签名校验需要较大的栈空间

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp

; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
```

> ⚠ 警告
>
> 请特别注意堆栈大小，要调大，否则将无法执行签名校验，返回内存不足的错误信息。

3.3.2.2.2 cortex_chipid_binding.c

☑ 更换公钥为ICWKEY 导出的公钥

```
//使用ICWKEY 中的 公钥进行替换
const static uint8_t PUBLIC_KEY[49]={

0x04,0x00,0x7F,0xFE,0xF3,0x5A,0xFB,0x48,0xC3,0xEB,0xE8,0xE5,0x41,0xDE,0xAF,0x99,

0x89,0x48,0x8C,0x31,0x93,0x2A,0x91,0x81,0xD1,0x17,0x62,0xA5,0x89,0xA6,0x77,0x02,

0x14,0x60,0xC7,0x79,0x1E,0x33,0xDF,0x8F,0xE0,0xF0,0xC2,0x47,0x03,0x49,0x7B,0x5F,
    0xF7
};
```

3.3.2.2.3 cortex_chipid_binding.h

☑ 填写ID 地址（请看提示信息）

☑ 修改签名地址为PowerWriter 中签名地址

☑ 根据实际情况，是否开启占位符。

```
/* Exported define
-----------------------------------------------------------*/
/* The following parameter definitions must be consistent with the
actual chip and burner settings */
```

> **提示**
>
> UID_CHIP_ADDR 的地址，可以用过PowerWriter 选择签名模式为 Matrix，导出源码中可以看到实际的ID地址。

3.3.2.2.4 main.c

- [x] 初始化ID
- [x] 验证签名

```
/* Private user code
---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
//用于串口打印日志信息
int fputc(int ch, FILE *f)
{
        uint8_t ch8 = (uint8_t)ch;
        HAL_UART_Transmit(&huart2,&ch8,sizeof(ch8),5);
        return (ch);
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */


  /* MCU
Configuration---------------------------------------------------------*/
```

> 提示
>
> 示例代码只是演示，为了更加安全，请注意隐藏代码，可提高安全性，有必要时，联系我们获取 **MCU 通用安全保护库**，来进一步提升固件安全，防止固件被逆向反编译、破解、和修改。

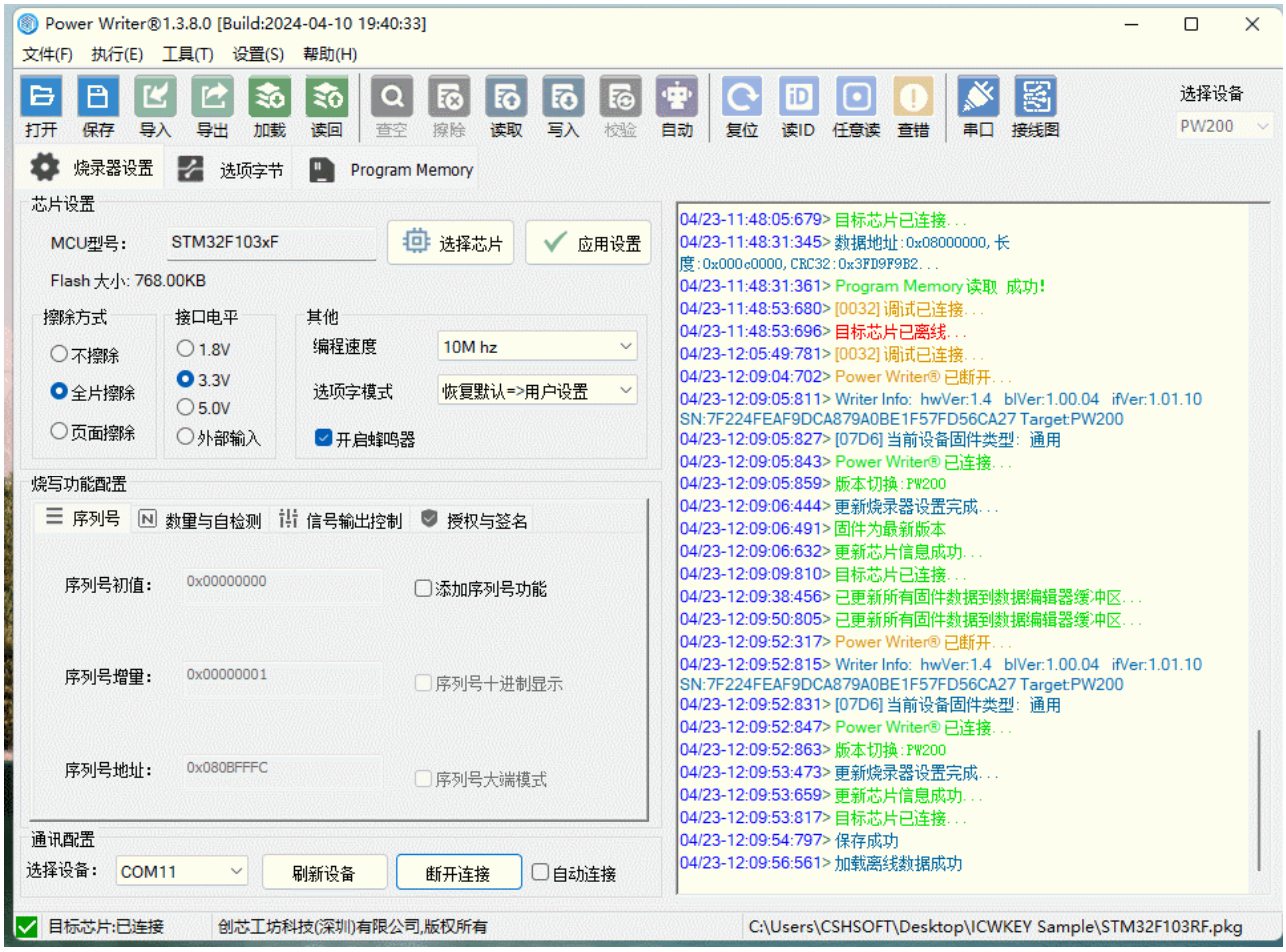
3.3.2.3 编译

```
#define SISSDK_LOG_ENABLE       //disbale /Enable   #warning You have to implement fput functions to use log print function
```

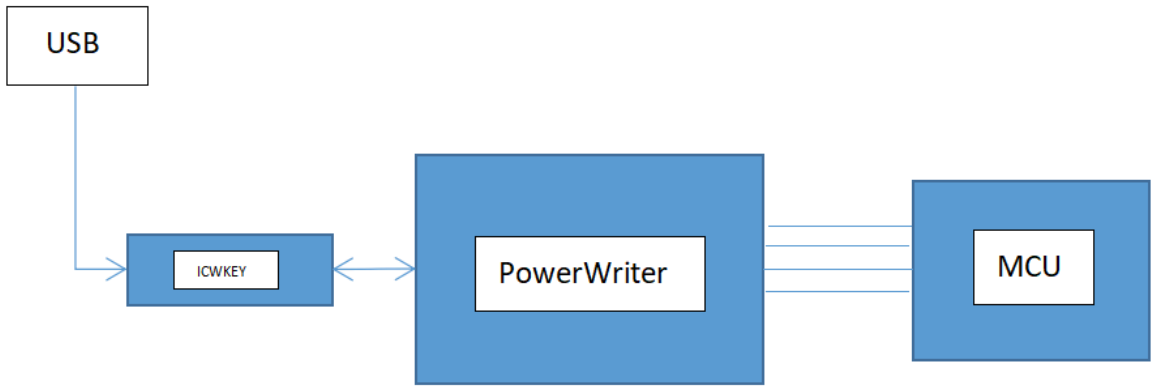
验证时可开启日志，方便查看结果，编译项目，将在目录 Output\ TargetIC_Example.bin 生成测试固件。

3.3.2.4 验证

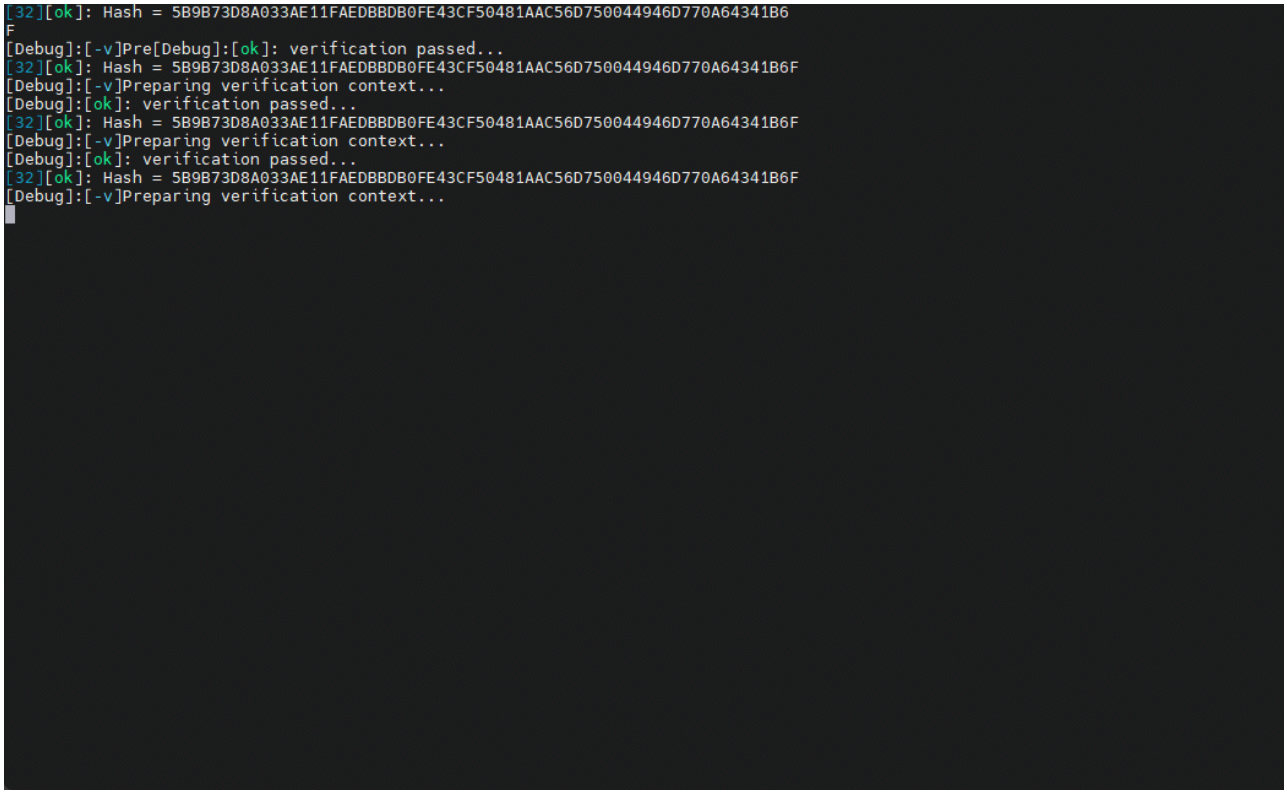
重新打开PowerWriter 项目，在Program Memory 页面添加 TargetIC_Example.bin 测试固件，并将项目加载到PowerWriter设备，如下所示：



连接ICWKEY 到PowerWriter， 并连接需要编程的MCU 目标PCB板， 并连接电源进行编程，参考接线如下所示:



编程结束后，连接目标PCB 的串口TX 引脚，即可看到输出的签名验证信息，参考如下：

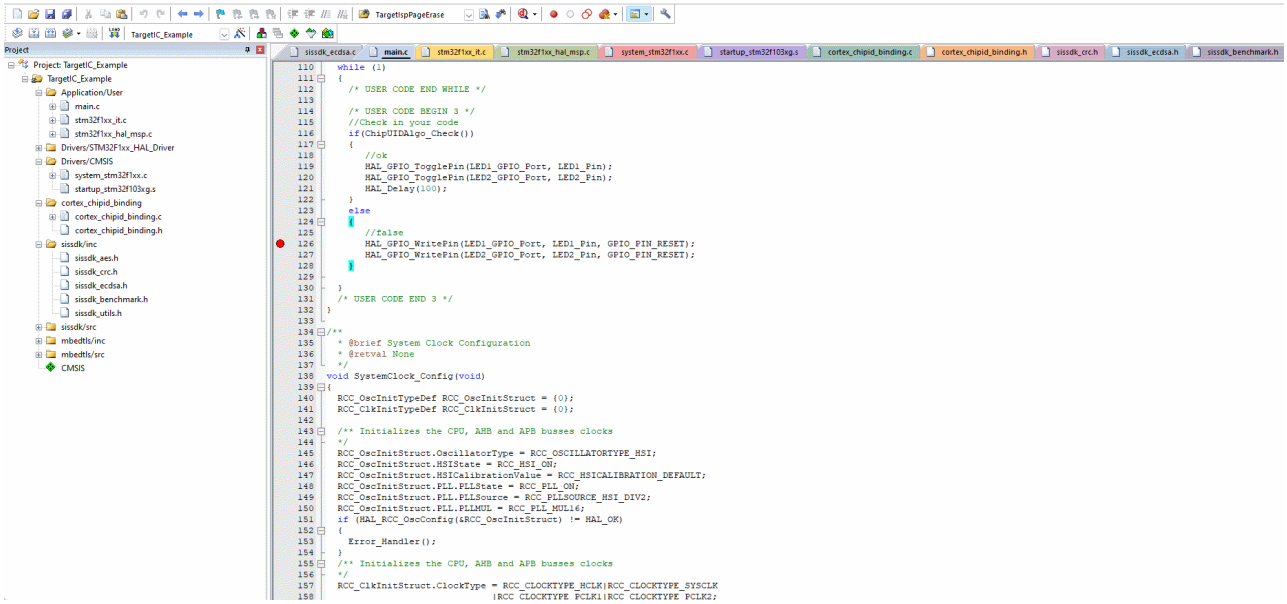


3.3.2.5 调试方法

使用PowerWriter 对目标固件进行签名并编程到目标芯片之后，可通过设定的状态输出，来检查签名是否生效，在复杂的场景下，单纯看工作状态无法判定出现问题的位置，此时，需要对目标芯片进行调试，调试步骤如下：

- 参考编译验证流程，完成编程
- IDE选择：**不擦除目标芯片**，**不编程目标芯片**，**不校验目标芯片**的方式进行。

参考演示如下所示：


TargetIspPageErase
TargetIC_Example
Project
Project: TargetIC_Example
TargetIC_Example
Application/User
main.c
stm32f1xx_it.c
stm32f1xx_hal_msp.c
Drivers/STM32F1xx_HAL_Driver
Drivers/CMSIS
system_stm32f1xx.c
startup_stm32f103xg.s
cortex_chipid_binding
cortex_chipid_binding.c
cortex_chipid_binding.h
sissdk/inc
sissdk_aes.h
sissdk_crc.h
sissdk_ecdsa.h
sissdk_benchmark.h
sissdk_utils.h
sissdk/src
mbedtls/inc
mbedtls/src
CMSIS
sissdk_ecdsa.c
main.c
stm32f1xx_it.c
stm32f1xx_hal_msp.c
system_stm32f1xx.c
startup_stm32f103xg.s
cortex_chipid_binding.c
cortex_chipid_binding.h
sissdk_crc.h
sissdk_ecdsa.h
sissdk_benchmark.h
while (1)
{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
//Check in your code
if(ChipUIDAlgo_Check())
{
//ok
HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
HAL_Delay(100);
}
else
{
//false
HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
}
}
/* USER CODE END 3 */
}
/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
/** Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
Error_Handler();
}
/** Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

编辑本页

版本：**Next**

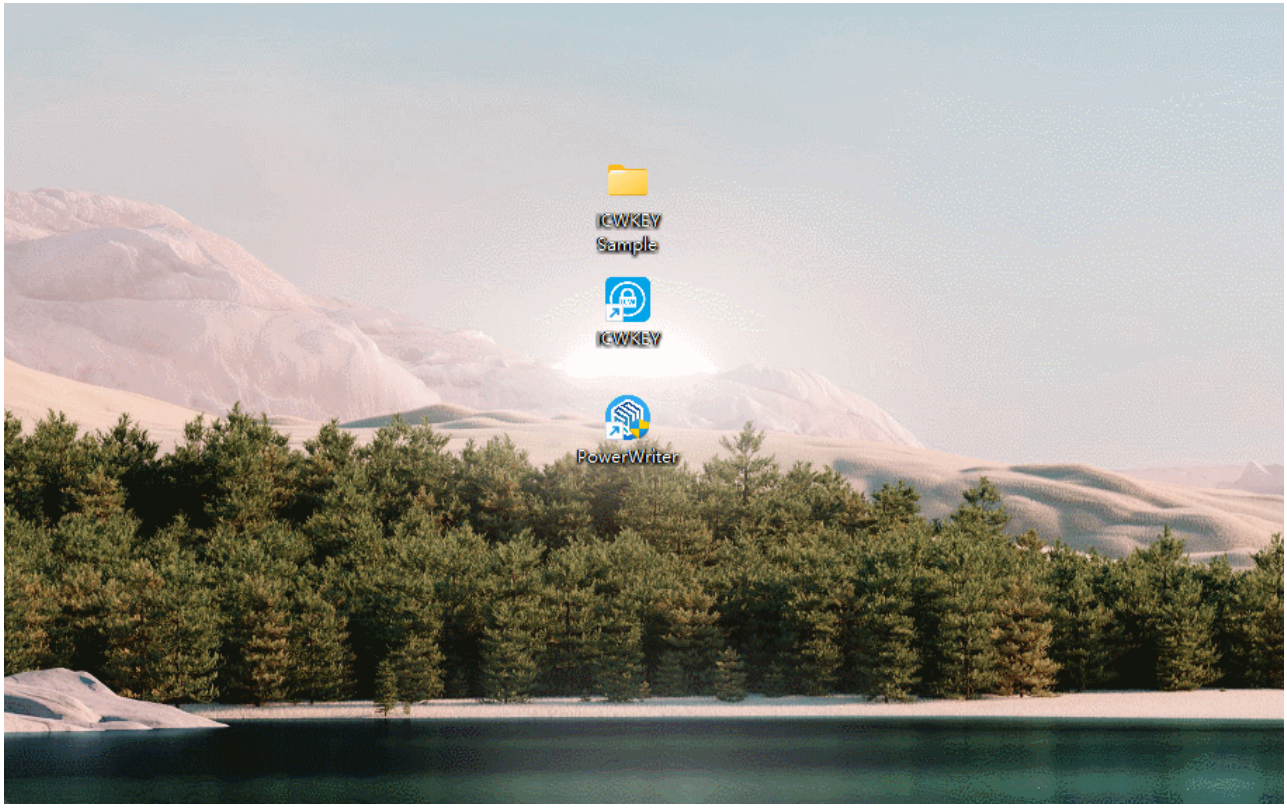
3.4 Matrix示例

3.4.1 准备

Matrix 签名为一种简易的校验算法，ICWKEY(PowerWriter) 随机生成一一种组合，来对ID进行加密，然后将加密的信息在生产时写入到目标芯片，目标芯片启动时，对签名进行校验，从来用来验证，当前芯片是否写入签名信息来进行固件保护的方法，在开始之前，我们需要按照流程核查所有的准备工作都已经完成。

- ☑ PowerWriter 项目中使用ICWKEY 签名(或者 ICWKEY 签名锁定模式)。
- ☑ 已经设置签名地址（如0x08002000）。
- ☑ PowerWriter 端的通讯信息已经同步到ICWKEY 的项目中，并与项目重新建立加密通讯。
- ☑ 合理设置可授权次数，比如设置为10000。
- ☑ 签名方法：选用**Matrix**签名，保存到ICWKEY，并导出了源码。

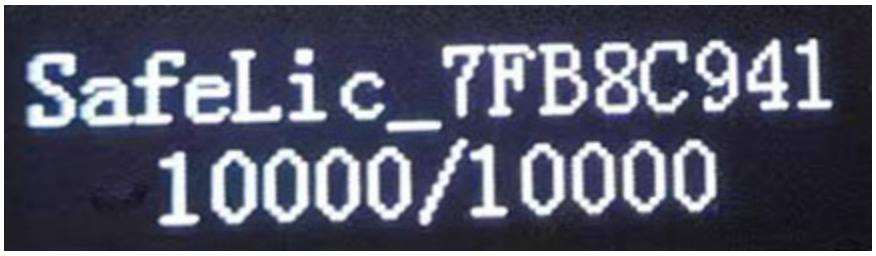
演示如下所示：



如以上步骤均完成，则可以看到ICWKEY 设备的显示信息，以及导出的源码信息，参考如下：

STM32F103RF.pkg	2024/4/23 14:25	PKG 文件	47 KB
SafeLic_7FB8C941.uprj	2024/4/23 10:19	UPRJ 文件	1 KB
SafeLic_7FB8C941 - matrix.uprj	2024/4/23 15:06	UPRJ 文件	1 KB
cortex_chipid_binding.h	2024/4/23 15:06	C Header 源文件	6 KB
cortex_chipid_binding.c	2024/4/23 15:06	C 源文件	8 KB

同时ICWKEY 的设备将显示如下的信息：



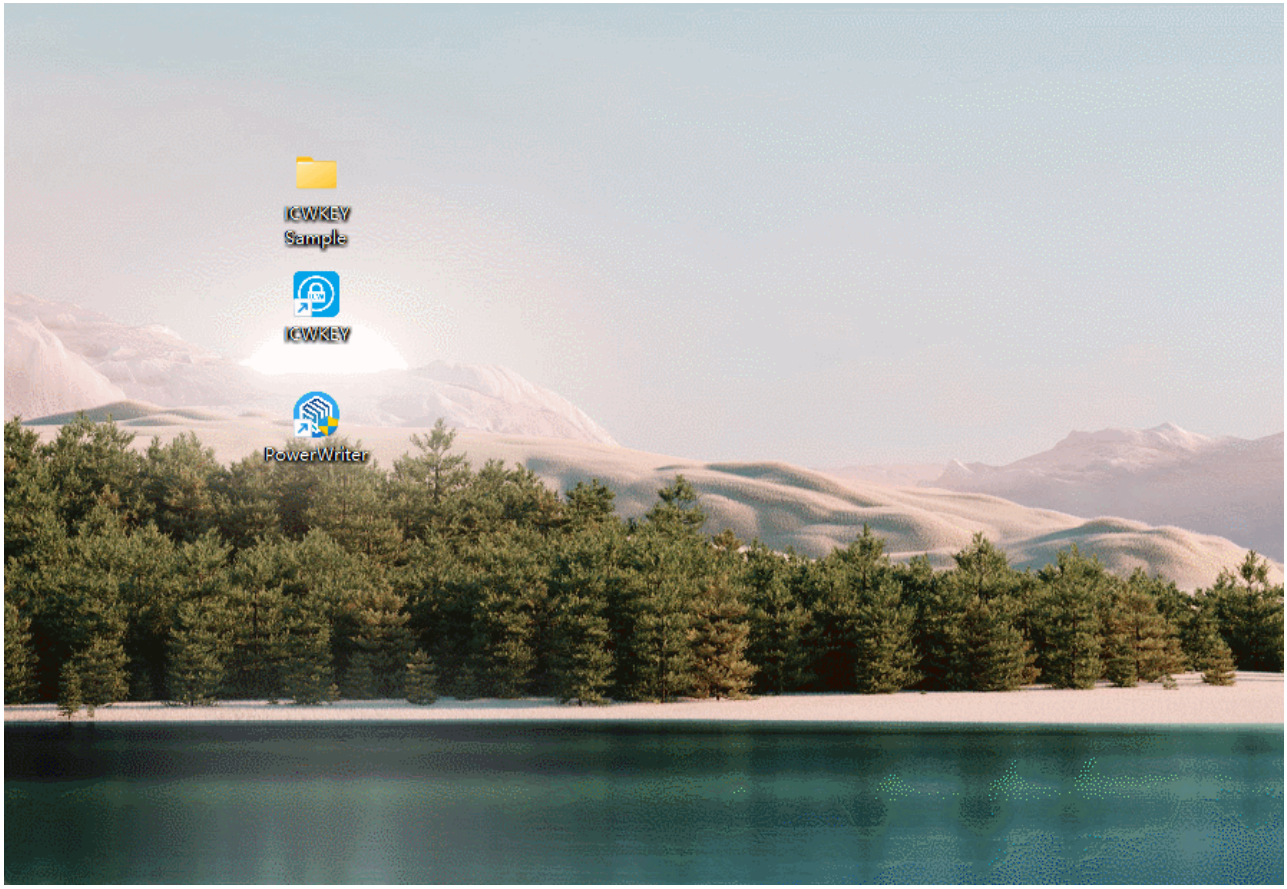
3.4.2 示例工程

3.4.2.1 准备

示例工程路径ICWKEY 的安装路径下，具体为：

```
C:\Users\用户名\AppData\Local\ICWKEY\Examples_for_mdk
```

可通过ICWKEY 桌面图标，快速定位到为止，并拷贝 **Matrix** 示例工程到指定路径，并解压，参考演示如下：



3.4.2.2 代码结构

3.4.2.2.1 cortex_chipid_binding.c

☑ 更换ICWKEY 导出的函数

```
//使用ICWKEY 中的导出的代码进行替换
//The following code may warn in KEIL(MDK), ignore it
static void ChipUIDAlgo(char pUserID[], char pChipID[], char pKey[])
{
    pKey[0] = pChipID[8] * pChipID[3] | pUserID[8] & pChipID[10] ;
    pKey[1] = pChipID[5] + pChipID[2] - pChipID[7] ^ pChipID[11] ;
    pKey[2] = pChipID[9] - pUserID[4] ^ pUserID[2] * pUserID[10] ;
    pKey[3] = pChipID[1] | pUserID[0] & pUserID[11] + pUserID[7] ;
```

3.4.2.2.2 cortex_chipid_binding.h

- [x] 填写ID 地址（请看提示信息 。
- [x] 修改签名地址为PowerWriter 中签名地址 。
- [x] 替换为ICWKEY 中导出的UID_USERID_KEYx 。
- [x] 根据实际情况，是否开启占位符。

```
/* Exported define
-----------------------------------------------------------*/

/* The following macros are automatically exported by the software
supporting the burner.
   Please do not modify them to keep them consistent */

#define    UID_CHIP_MASK                0x5BD489F0              //Random
generation
#define    UID_CHIP_SIZE                12                      //ChipID
Size
/* 目标芯片的ID地址，可根据芯片手册查询 */
#define    UID_CHIP_ADDR                (0x1FFFF7E8^UID_CHIP_MASK)
//ChipID Inner Addr in chip

#define    UID_KEY_LENGTH               12                       //The
password is the same length as the user ID input
//签名信息存储地址，改为PowerWriter项目中签名信息的存储地址 0x08002000
#define    UID_KEYADDR_INNER            (0x08002000^UID_CHIP_MASK) //Key
Store Addr In flash

//替换为 ICWKEY 中导出的 密码
#define    UID_USERID_LENGTH            UID_KEY_LENGTH
//Customize password length
#define    UID_USERID_KEY1              0xAD683BDB
//UserID 1
```

> **提示**
>
> UID_CHIP_ADDR 的地址，可以用过PowerWriter 选择签名模式为 Matrix，导出源码中可以看到实际的ID地址。

3.4.2.2.3 main.c

- [x] 初始化ID
- [x] 验证签名

```
/* Private user code
---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
//用于串口打印日志信息
int fputc(int ch, FILE *f)
{
        uint8_t ch8 = (uint8_t)ch;
        HAL_UART_Transmit(&huart2,&ch8,sizeof(ch8),5);
        return (ch);
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */


  /* MCU
Configuration---------------------------------------------------------*/
```

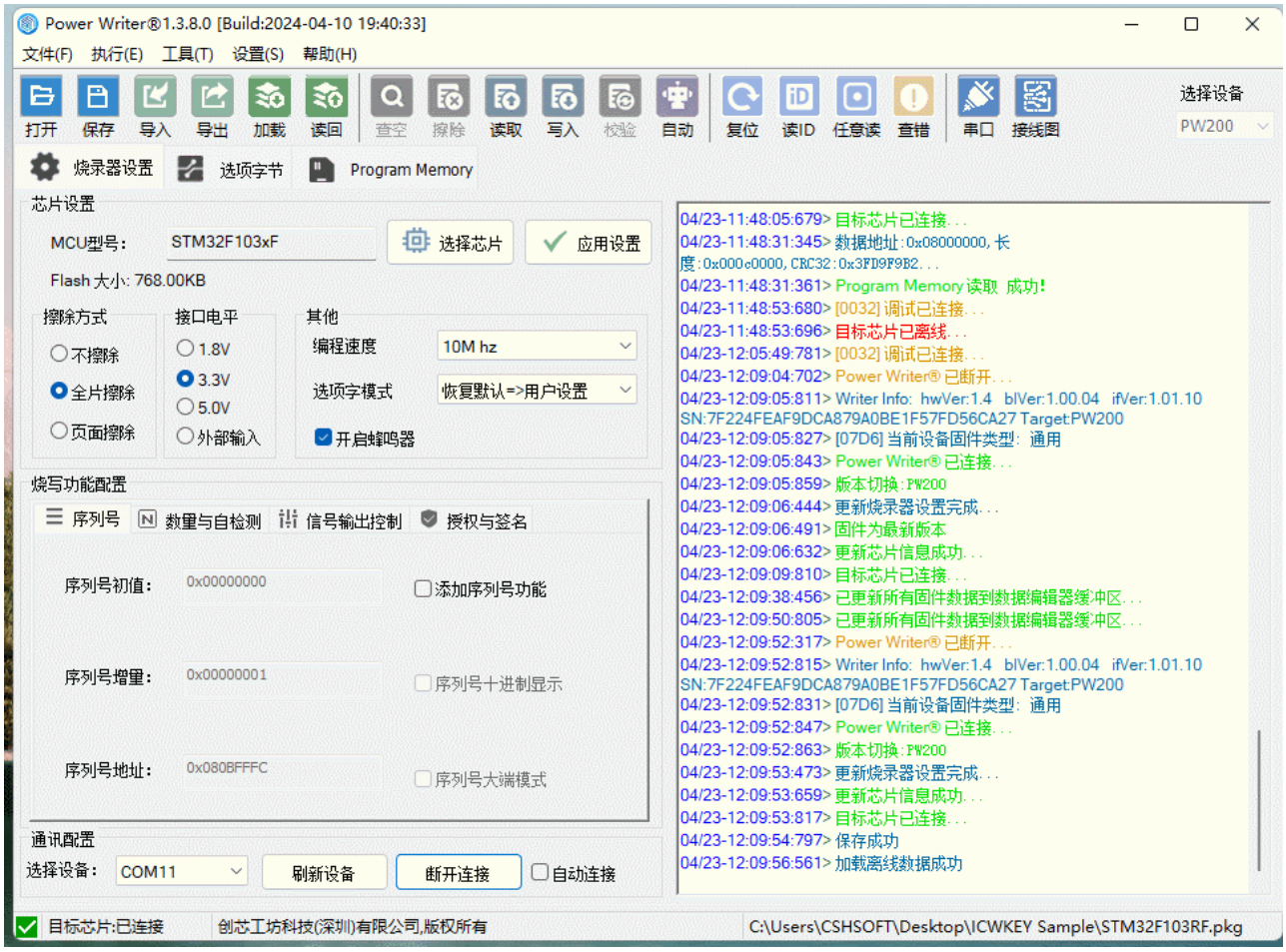
> **提示**
>
> 示例代码只是演示，为了更加安全，请注意隐藏代码，可提高安全性，有必要时，联系我们获取 **MCU 通用安全保护库**，来进一步提升固件安全，防止固件被逆向反编译、破解、和修改。

3.4.2.3 编译

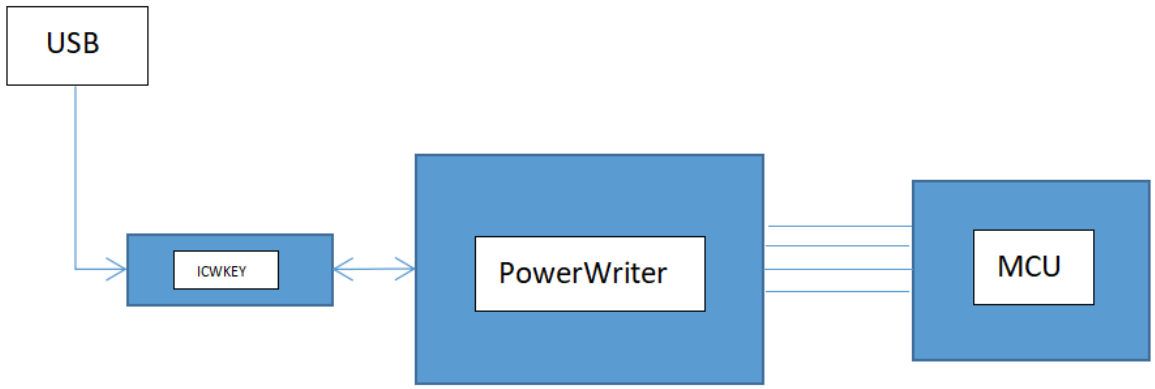
编译项目，将在目录 Output\ TargetIC_Example.bin 生成测试固件。

3.4.2.4 验证

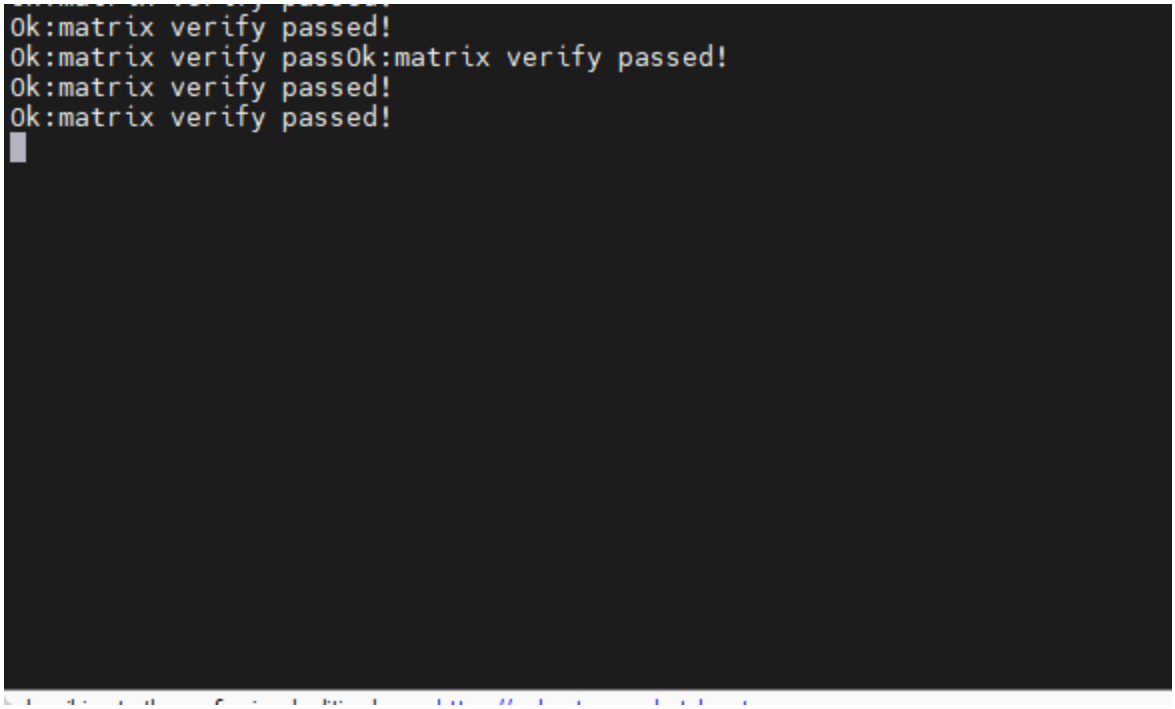
重新打开PowerWriter 项目，在Program Memory 页面添加 TargetIC_Example.bin 测试固件，并将项目加载到PowerWriter设备，如下所示：



连接ICWKEY 到PowerWriter， 并连接需要编程的MCU 目标PCB板， 并连接电源进行编程，参考接线如下所示:



编程结束后，连接目标PCB 的串口TX 引脚，即可看到输出的签名验证信息，参考如下：

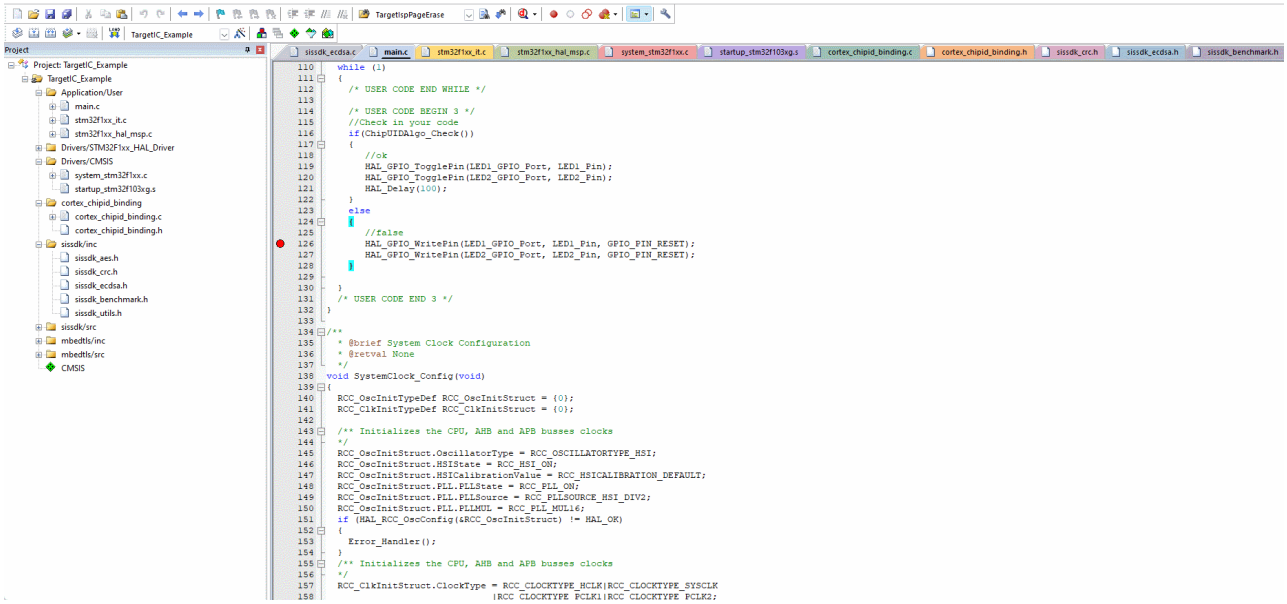


3.4.2.5 调试方法

使用PowerWriter 对目标固件进行签名并编程到目标芯片之后，可通过设定的状态输出，来检查签名是否生效，在复杂的场景下，单纯看工作状态无法判定出现问题的位置，此时，需要对目标芯片进行调试，调试步骤如下：

- 参考编译验证流程，完成编程
- IDE选择：**不擦除目标芯片**，**不编程目标芯片**，**不校验目标芯片**的方式进行。

参考演示如下所示：


TargetIspPageErase
TargetIC_Example
Project
Project: TargetIC_Example
TargetIC_Example
Application/User
main.c
stm32f1xx_it.c
stm32f1xx_hal_msp.c
Drivers/STM32F1xx_HAL_Driver
Drivers/CMSIS
system_stm32f1xx.c
startup_stm32f103xg.s
cortex_chipid_binding
cortex_chipid_binding.c
cortex_chipid_binding.h
sissdk/inc
sissdk_aes.h
sissdk_crc.h
sissdk_ecdsa.h
sissdk_benchmark.h
sissdk_utils.h
sissdk/src
mbedtls/inc
mbedtls/src
CMSIS
sissdk_ecdsa.c
main.c
stm32f1xx_it.c
stm32f1xx_hal_msp.c
system_stm32f1xx.c
startup_stm32f103xg.s
cortex_chipid_binding.c
cortex_chipid_binding.h
sissdk_crc.h
sissdk_ecdsa.h
sissdk_benchmark.h
while (1)
{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
//Check in your code
if(ChipUIDAlgo_Check())
{
//ok
HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
HAL_Delay(100);
}
else
{
//false
HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
}
}
/* USER CODE END 3 */
}
/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
/** Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
Error_Handler();
}
/** Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

编辑本页

版本：**Next**

4：参考指南

4.1 菜单功能

ICWKey 菜单功能详解

4.2 通讯设定

ICWKey 通讯设定

4.3 项目配置

ICWKey 详细项目配置方法

4.4 日志栏

ICWKey 日志栏介绍

版本：Next

4.1 菜单功能

4.1.1 文件

4.1.1.1 新建项目

新建ICWKEY 项目，点击新建项目按钮后，将重置当前得所有设置，如有数据需要保存，请提前保存数据，避免数据丢失。

4.1.1.2 加载项目

加载ICWKEY 项目文件，后缀为 **uprj**，点击加载项目，将弹出加载项目对话框，填写项目密码，浏览项目路径，然后点击确定按钮，项目将被加载。

4.1.1.3 保存项目

保存当前得更改到项目文件。

4.1.1.4 项目另存为

将项目另存为新项目。

4.1.1.5 退出

退出ICWKEY 软件。

4.1.2 执行

4.1.2.1 默认通讯设定

通过此功能，可将通讯设定得 项目名称、密码、向量等信息恢复为如下默认值。

```
/*
默认通讯设定
名称：ICWorkshop
密码：30313233343536373839414243444546
向量：46454443424139383736353433323130
*/
```

4.1.2.2 项目通讯设定

如之前将通讯设定改为了默认通讯设定，通过此功能，恢复通讯连接信息为当前加载项目得设置。

4.1.2.3 保存项目并更新

此功能将保存项目，并将最新得项目同步到 ICWKEY 硬件设备。

4.1.3 帮助

4.1.3.1 官方网站

访问创新工坊官方网站、PowerWriter 官方网站 www.powerwriter.com

4.1.3.2 许可协议

查看用户协议

4.1.3.3 用户手册

查看ICWKEY 用户手册离线 PDF。

4.1.4 语言

4.1.4.1 简体中文

设置为简体中文。

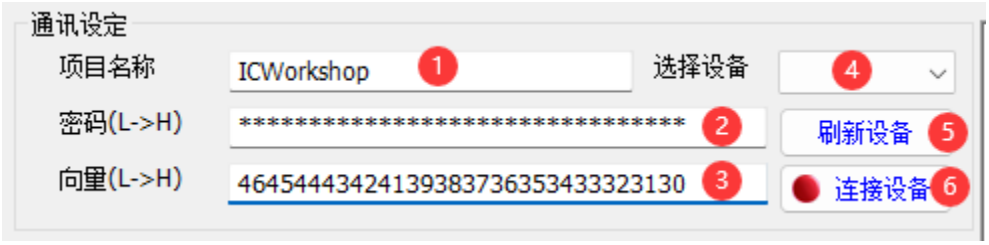
4.1.4.2 英语

设置为英文。

编辑本页

版本：Next

4.2 通讯设定



- **项目名称**：默认为ICWorkshop。
- **密码**：默认为 30313233343536373839414243444546。
- **向量**：默认为 46454443424139383736353433323130。
- **设备列表**：当前识别到ICWKEY 设备列表。
- **刷新设备**：刷新设备列表。
- **连接设备**：连接选择得设备。

请参考 演示->同步项目名称到ICWKEY。

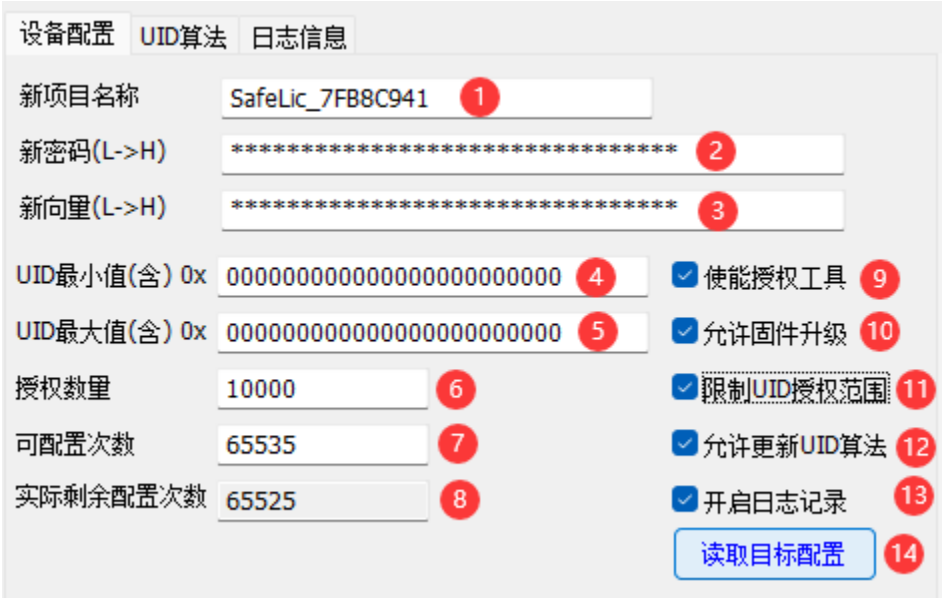
编辑本页

版本：Next

4.3 项目配置

4.3.1 设备配置

设备配置页面，包含签名设置所需得大部分配置，请看详细标注信息，如下所示：



- **新项目名称**：新项目名称，此字段从PowerWriter项目中复制。
- **新密码**：新密码，此字段从PowerWriter项目中复制。
- **新向量**：新向量，此字段从PowerWriter项目中复制。
- **UID 最小值**：限定UID最小值。
- **UID 最大值**：限定UID最大值。
- **授权数量**：控制实际可授权得数量。
- **可配置次数**：当前ICWKEY 设备可以更改配置的次数，默认为 65536，每配置一次，计数器 -1，当为0时，将不再可以进行修改！

- **实际剩余配置次数**：当前设备剩余可修改次数。
- **使能授权功能**：开关授权功能。
- **允许固件升级**：是否允许固件升级。
- **限制UID授权范围**：限制UID 的使用范围。
- **允许更新UID算法**：是否允许更新授权算法。
- **开启日志记录**：记录授权日志信息，常用于做报表。
- **读取目标配置**：读取当前设备的配置信息。

> ⚠ 警告
>
> - 新项目名称、密码、向量从PowerWriter项目中复制、更新之后，请保存好项目，避免丢失，否则会导致ICWKEY 无法连接。
> - UID 设置，只有在开启限制UID 授权范围时有效。
> - **可配置次数**：请注意此位置非授权数量，而是设备可以更改设置的次数，当次数为0时，设备将锁定，无法再重复修改，除非必要，请勿修改此信息。

4.3.2 UID算法

当前自带两种签名算法，第一种时Matrix 签名，为一种随机矩阵加密算法，优点是占用资源极少，可对目标芯片进行签名与校验，防止固件被直接拷贝使用，第二种是ECDSA 数字签名，为一种非堆成电子签名算法，此算法目前加密强度极大，算法本身难以破解，但是仍需进一步增强代码本身的防护，防止签名被移除，可集成 **MCU 通用高级软件保护库** 来提升固件的安全性，如有需要，请联系 cs@icworkshop.com。

4.3.2.1 Matrix

此处设置一般保留默认设置即可，可随机生成Key。

随机生成Matrix 信息，操作完成后，点击导出源码（需集成到固件中），最后点击编译并保存，更新设置。

> **提示**
>
> 设置完成后，请一定要**点击保存按钮**，避免设置未更新，如忘记操作，可重新执行一遍操作，并导出源码。

4.3.2.1 ECDSA

随机生成：生成ECDSA 数字签名公私密钥对。

导出源码：导出源码信息，用于集成。

编译并保存：操作完成后，保存设置，并更新。

> **提示**
>
> 设置完成后，请一定要**点击保存按钮**，避免设置未更新，如忘记操作，可重新执行一遍操作，并导出源码。

4.3.3 日志信息

授权总数：当前设备配置的授权数量总计。

已用次数：当前请求授权的次数累计。

成功次数：已成功分发证书的次数累计。

失败次数：分发失败的次数总计。

未知错误：未知错误，一般归类为失败次数。

测试授权：

填写ID 信息，将生成当前芯片的授权信息，常用于调试。

> **提示**
>
> - ECDSA 授权测试，每次返回的签名信息，不一定相同，请知悉。
> - 失败次数 + 成功次数 + 未知错误 = 已用次数。

编辑本页

版本：Next

4.4 日志栏

ICWKEY 日志栏显示当前的操作流程，以及动作的操作结果，格式如下所示

```
==========================================
            创芯工坊安全授权盾(ICWKEY)快速指南

在正式使用创芯工坊安全授权盾(ICWKEY)之前,我们建议您先阅读一下
创芯工坊安全授权盾(ICWKEY)用户手册,可以通过菜单->帮助->用户手册
打开用户手册文档。

创芯工坊科技（深圳）有限公司
网址：https://www.icworkshop.com
电话：400-1568-598
邮箱：cs@icworkshop.com
==========================================
04/23-17:01:57:337> 加载成功:SafeLic_7FB8C941 - matrix.uprj
04/23-17:08:43:222> 加载成功:SafeLic_7FB8C941 - matrix.uprj
04/23-17:10:17:002> 没有可连接的设备
04/23-17:10:23:800> 保存成功:SafeLic_7FB8C941 - matrix.uprj
04/23-17:33:55:941> 加载成功:SafeLic_7FB8C941 - matrix.uprj
04/23-17:33:57:946> 没有可连接的设备
04/23-17:34:05:292> 侦测到有设备插入
04/23-17:34:06:833> 启动设备配对
04/23-17:34:06:998> 配对成功
04/23-17:34:07:021> 成功读取目标设备配置，项目代号:SafeLic_7FB8C941,序列
号:21B53974DE21BA1179EF54CA853E89DE,HW版本:v1.00,FW版本:v1.03,授权总
数:10000,剩余授权数量:9999,剩余可配置次数:65525,UID算法:向量矩阵加密(Matrix)
04/23-17:58:55:311> 成功读取目标设备配置，项目代号:SafeLic_7FB8C941,序列
号:21B53974DE21BA1179EF54CA853E89DE,HW版本:v1.00,FW版本:v1.03,授权总
数:10000,剩余授权数量:9999,剩余可配置次数:65525,UID算法:向量矩阵加密(Matrix)
04/23-17:58:59:779> 发送chipID
```

编辑本页

版本：**Next**

5：特别说明

5.1 常见问题

ICWKey 常见问题

5.2 注意事项

ICWKey 注意事项

版本：Next

5.1 常见问题

1：USB 驱动安装失败

如果电脑操作系统是Windows XP、Win7 或 Win8，且非官方原版， 而是精简系统，可能会遇到安装失败的问题，可以搜索 “**ST 虚拟串口驱动安装失败**”获取解决方案，不要选择与实际系统的不相符的驱动进行安装，必要时联系技术支持。

2：ICWKEY连接判断

确保ICWKEY客户端 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。

3：验证签名数据

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。

4：调试时失败

签名信息由PowerWriter进行编程，调试时没有授权数据，所以校验不能通过，请参考 ECDSA 范例 和 Matrix 范例。

5：签名数据长度

- Matrix : 长度一般为 12 字节，跟随设置，可设置 4 字节、8字节、12字节。
- ECDSA：长度不超过 141 字节。

6：编程失败原因

- 接线问题： 接线错误， 接线松动。
- 配置出错： 选择的芯片和目标芯片不一致。
- 次数耗尽：PowerWriter 或 ICWKEY 设置的次数用完。
- 禁用烧录：目标芯片已经关闭了烧录功能，比如开启了二级读保护， JTAG &serial wire 已经被禁用。
- 引脚复用：烧录IO 被程序改为普通IO， 可尝试连接RESET引脚，使用under reset 模式来进行烧录。
- 供电不足： 电压过低可能会导致烧录失败。

7：编程后无法重新编程

可能是目标芯片已经开启了读保护。

版本：Next

5.2 注意事项

1：源码修改

ECDSA 签名模式下，通常需要修改以下信息：

- 更换公钥为ICWKEY 导出的公钥。
- 填写ID 地址（请看提示信息）。
- 修改签名地址为PowerWriter 中签名地址。
- 根据实际情况，是否开启占位符。

ECDSA 签名模式下，通常需要修改以下信息：

- 更换ICWKEY 导出的函数。
- 填写ID 地址。
- 修改签名地址为PowerWriter 中签名地址 。
- 替换为ICWKEY 中导出的UID_USERID_KEYx 。
- 根据实际情况，是否开启占位符。

具体请参考 ECDSA 范例 和 Matrix 范例。

2：项目密码

请牢记要保存PowerWriter 项目和 ICWKEY 的项目文件，项目文件丢失后，可能造成无法正确进行签名配置，或者连接到ICWKEY 设备。

3：签名地址

签名地址的存放，请不要超出Flash 的空间，避免造成无法烧录，同时，请勿和代码进行重叠，如担心会造成重叠，请开启占位符，开启占位符之后，会在固件中预留空间，避免出现重叠现象，同时将签名地址尽可能地址靠前。

3：占位符

开启时会在固件中预分配空间，避免出现覆盖固件数据，不开启时，则指定的签名地址处的数据会被覆盖，覆盖的长度参考

签名数据长度