创芯工坊

Power Writer 系列

用户参考手册(RM0001)

扫码关注微信公众号

官方网站

创芯工坊硬件平台技术部

创芯工坊 **Power Writer** 系列

用户参考手册**(RM0001)**

发布信息

本文档做了以下更改

更改历史

版本	日期	修正日志
V1.00	2020/3/23	第一版本发布
V1.01	2020/06/30	更新新功能描述，以及修复部分编辑错误

1. 特别注意

本文件中的资料如有更改，恕不另行通知

本文档原样提供给用户，创芯工坊不对手册中的编译遗漏或者编辑错误、阅读本手册时遇到障碍而导致造成生产方面损失负责，如有任何疑问请及时联系官方市场、客服或技术支持团队，请留意手册上的“联系我们”章节内容。

本文件受版权和其他相关权利以及所包含信息的实践或实施的保护，为了保证本文档的准确性和可维护性，未经过容许禁止复制、复印、翻译、修改本文档的全部或者部分内容，或者将其转化为其他电子文档格式或者是机读形式。

本文档的功能可能在产品更新迭代中,和实际的产品会存在一定的差异,请以实际产品为准。我们会努力提供最新的产品资料，如用户手册、Demo、固件、配套的驱动、软件以及相关 SDK。

保密状态

本文件是非机密的，您的任何使用均受您与创芯工坊之间的协议条款约束。

产品状态

本文件中的信息包含已经发售的版本，但也包含一些关于未来产品规划的信息,请以官方发布信息为准。

网站地址

https://www.icworkshop.com

1：前言

1.1. 关于本文档

本文档作为 Power Writer 系列产品用户的官方参考手册

1.1 缩写

ICW	**ICWORKSHOP** 创芯工坊
读/写(rw)	执行读写操作
读(r)	执行读操作
写(w)	执行写操作
保留(Res.)	保留
SOC	System-on-chip，片上系统
ICWKEY	创芯工坊安全授权硬件模块

1.2 术语

ADI	ARM 调试访问端口
Aligned	存储在地址中的数据项，该地址可以被定义其数据大小的字节数整除。对齐双字、字和半字的地址分别可以被 8、4 和 2 整除。一个对齐访问是指访问的地址与访问的每个元素的大小保持一致。
Big-endian	在数据的上下文中，Big-endian 被定义为内存组织一个字的有效字节的地址低字节数据存放在地址高位，例如: •字对齐地址处的字节或半字是该字中最重要的字节或半字地址。 •半字对齐地址处的字节是该地址处的半字中最重要的字节。 参见 Little-Endian 和 Endiannes
Debugger	一种调试系统，包括一个程序，用于检测、定位和纠正软件故障支持软件调试的自定义硬件。
Doubleword	64 位数据项。在 Arm 芯片中，双字通常至少与字对齐
Endiannes	在一个较大的数据结构中，当该结构存在时，确定连续数据字节顺序存储在内存中的方案。 参见 Little-Endian 和 Big-Endian。

Halfword	一个 16 位的数据项。半字在 Arm 系统中通常是半字对齐的
JTAG	一个 IEEE 小组专注于硅芯片测试方法。许多调试和编程工具都使用联合测试动作组(JTAG)接口端口，用于与处理器通信。 参见 IEEE Std 1149.1-1990《IEEE 标准测试访问端口和边界扫描体系结构规范》，来自 IEEE 标准协会。
JTAG-AP	DAP 的一个可选组件，提供对片上扫描链的调试器访问
JTAG-DP	DAP 的可选外部接口，提供用于调试访问的标准 JTAG 接口
Little-Endian	在 Arm 架构的上下文中，little-endian 被定义为内存组织，其中大部分是字的有效字节位于比最低有效字节更高的地址。
SWD	使用 SOC 和调试器之间的串行连接的调试实现。这个连接正常需要一个双向数据信号和一个单独的时钟信号，而不是 JTAG 所需的 4 到 6 个信号连接。
SW-DP	串行线调试接口。
SWJ-DP	SWJ-DP 是 JTAG-DP 和 SW-DP 的组合，您可以使用它们来连接串行线路调试(SWD)或者 JTAG 探测到目标芯片。
Flash-Tool	用于对目标芯片进行烧录写入的相关软件和硬件的统称

2 ： 概览

2.1 创芯工坊介绍

图 2.1.1 创芯工坊官网

创芯工坊科技（深圳）有限公司以“创新驱动产业，融合创造价值”为主旨，致力于以“集成电路+互联网”的新模式，打造精细化的产业互联网平台。旗下创芯工坊平台(https://www.icworkshop.com) 基于互联网技术，同时融合云存储、云烧录及网络加密技术，运用电子商务运营模式，将传统芯片程序交付模式转变为在线交付，实现方案交付的软硬分离，极大程度保护了开发者的知识产权权益，同时提高了方案交付效率

图 2.1.2 创芯工坊核心技术概览

2.2 创芯工坊系统架构

下图是创芯工坊的简易系统框图

图 2.2.1 创芯工坊平台简易流程

创芯工坊平台拥有完善的远程量产控制，丰富的芯片品种，以及丰富的烧录器支持。随着研发的不断投入，创芯工坊官方烧录器 Power Writer 也正式发布，官方烧录器除了包含市面常见烧录器的基础功能之外，重点在联网控制、数据安全上都有了质的提升，保证用户固件和配置数据不会被盗取。

2.3 Power Writer 系列概述

2.3.1 Power Writer 解决的问题

创芯工坊互联网+集成电路生产管理模式

烧录器已经不单是一个烧录器，同时具备了 IoT 产品的属性，Wireless 版本将支持蓝牙（App/小程序）进行在线离线量产，支持直连服务器执行在线配置和权限控制（开发中）

用户对于量产过程中数据安全的担忧

Power Writer 在安全、权限控制上投入大量的研发，无论是 Power Writer 硬件本身、还是配套的相关软件、服务端、第三方开放授权服务器 API、在保护数据安全上都做到了首创。

授权控制

Power Writer 系列产品包含多种授权控制权限：

授权类型	描述
官方服务器在线授权	支持创芯工坊官方内建授权
第三方服务器在线授权	基于创芯工坊提供授权服务器 SDK 自建在线授权

Power Writer 内建离线授权	基于 Power Writer 内置的随机矩阵算法离线授权
ICWKEY 授权盾	创芯工坊官方单独非对称授权工具
ICWKEY 自定义授权	大客户可以使用 SDK 开发自定义授权模块

OEM 定制化生产需求

对于某些特定客户，如标准版的功能无法满足需求，创芯工坊也可提供产品定制服务。

2.3.2 Power Writer 外观一览

图 2.3.2 Power Writer 预览效果

2.3.3 Power Writer 简介

Power Writer 是创芯工坊官方推出的 Debugger + Programmer(Writer)，主要面对个人开发者和小批量生产、授权控制领域。随着产品规划地推进，将会推出更多不同类型产品来满足不同类型的需求。

产品参数：

- ✓ 产品尺寸：92.000mm * 56.000mm * 16.000mm
- ✓ 工作电压：DC5V
- ✓ 产品功耗：30mA@5V~100mA@5V

支持的芯片：

✓ STM32 全系列（已完成）
✓ STM8 全系列（适配中）
✓ 更多支持芯片持续适配中，请留意创芯工坊公众号发布的消息

2.3.4 Power Writer 特性简介

Power Writer 系列产品提供丰富的功能设定，让您无论在开发测试，还是实际量产、权限控制上都做到游刃有余，下面是功能特定的不完全列表，随着产品的不断升级、功能将会更加丰富。

2.3.4.1 特性列表概览 (安全部分)：

- 客户端软件采用 C++ 11 开发，而非其他厂商采用的是 C#/VB 等语言开发，虽然可以加密，但是通过一些技术手段可以获取到客户端软件源代码，导致后续的数据保护都成为一纸空谈。
- 客户端除了使用 C++ 开发之外，内部代码上采用了大量的检测技术，Hash/加密关键数据与代码，用于检测内存中的软件数据是否被读取或被篡改等异常操作，在每一个环节都确保用户数据的安全。
- 客户端软件采用商业保护软件，做了大量的逆向分析检测，多层防护。
- 客户端采用双证书 EV 代码签名，用户可以直观的判断软件来源是否来自官方版本，如果软件被篡改，用户将收到提示。
- 客户端支持多种加密算法、滚码算法、文件内部采用 Tag 标记和数据验证，而不是将密码存储在文件中，类似密码学中 zip 的加密方式。只有当密码输入正确时才能获取到真实数据，而大多密码长度为 16 字节以上。在代码开源的情形下，暴力破解都并非易事，确保用户数据的极致安全。
- 客户端软件到 Power Writer 的协议加密，采用非对称加密算法，在传输过程中不交换密钥。逆向分析人员只有同时破解了烧录器和 PC 端软件，并且取得加密的公钥和私钥时才能解密数据包。
- 客户端软件到 Power Writer 的协议同时做了动态滚码加密，在非对称 ECC 加密的基础上，多次发送同一个数据包，将得到不同的 packet。
- 客户端配合创芯工坊使用时，同时会经过服务端的验证，如果软件有任何修改，服务端将不会下发远程烧录数据。
- 通信协议签名校验，任何一帧数据都无法被篡改。
- 支持烧录目标芯片协议层加密。
- 支持烧录目标芯片动态加密、数据验证、读写防护和防注入。 (领先技术，一芯一密)
- 支持目标芯片写入读保护之后再写入部分数据，确保在烧录芯片数据烧录完但没有写入 Option Byte 时断开芯片，导致芯片会被读取的可能。(注：部分芯片不支持)
- Power Writer 硬件采用支持内存读取保护的芯片，支持 2 级保护的 L4 系列芯片，确保烧录器固件不被读取。
- Power Writer 固件内置绑定算法，每个 Power Writer 的固件都是独一无二的，升级固件时，由系统内部自动生成正确的固件。
- Power Writer 内置固件校验算法 Boot Loader、App 双验证。如果通过极端手段破解拿

到了烧录器固件，任何篡改都导致固件无法正确运行。

- Power Writer 将用户敏感数据存在在核心区域，并经过多重加密算法加密。
- Power Writer 每一台烧录器都有独立的 SN/OEM/并且支持和用户账户绑定 (开发中)。
- 支持服务器远程在线授权，可以通过创芯工坊内建的授权服务器对量产进行授权控制。
- 支持自三方自建服务器在线授权，通过采用创芯工坊授权服务器开发包，第三方用户可以快速搭建自己的授权服务器和自定义授权算法，自主控制产品的安全授权关键算法，创芯工坊作为平台提供方进行量产烧录，从根本上解决安全上的担忧。
- 支持内建智能矩阵离线授权，而非其他友商的固定加密算法（固定加密算法由于代码是一致的，导致反编译的代码有规律可循，给破解者可乘之机）。
- 支持 ICWKEY 非对称加密算法硬件授权算法，详见硬件授权模块的用户手册。
- 支持自定义 ICWKEY 的二次开发。(需取得创芯工坊授权)

图 2.3.4.1 Power Writer 数据安全防护一览图

2.3.4.2 特性列表概览 (功能部分)：

- 支持加密 Project
- 支持导入导出项目
- 支持加载加密项目到 Power Writer, 用于离线量产控制
- 支持从 Power Writer 读取离线项目。(读取需要项目密码,安全性所需)
- 支持在线芯片 Program Flash 区域读取
- 支持在线芯片 Program Flash 区域读取地址，读取大小设置，整片读取功能，方便用户读取目标芯片的固件数据
- 支持在线芯片查空,通过此项功能，用户可快速判断芯片是不是空片

- 支持在线擦除芯片，开发者、用户可直接通过软件擦除目标芯片
- 支持在线烧写 Program Flash 区域的数据，用户可通过在线编程的方式写入固件速度，而不是必须通过离线，或者 MDK/IAR/CUBEIDE 等 Debugger 方式写入
- 支持在线校验 Program Flash 数据
- 支持在线自动编程，自动执行擦除、写入、校验、更新 Option Byte 功能
- 支持在线复位目标芯片
- 支持设置读保护：可设置 Level-0,Level-1,Level-2 级别，并自动识别用户的保护位是否开启
- 支持完整的 Option Byte 设置列表
- 支持完整的 Option Byte 的默认设置
- Option Byte 支持多国语言
- 支持自动识别 Option Byte 中的选项,执行对应的操作
- 支持在线恢复出厂 Option Byte
- 支持在线读取 Option Byte
- 支持读取有锁芯片的选项字节
- 支持在线写入 Option Byte
- 支持保存用户自定义选项字节，用户可以将设置好的选项字节保存到文件，可以将其
- 发送给烧录厂或者是用于其他用途
- 支持加载用户自定义选项字节，可以从保存的选项字节加载到项目中
- Option Byte 动态实时同步，连接上目标芯片自动同步到 PC 客户端
- 支持 Bank 自动识别
- 支持多种 Option Byte 更新方式
  - 烧录前无操作 -> 烧录后无操作
  - 烧录前无操作 -> 烧录后写入用户自定义 Option Byte
  - 烧录前 Option Byte 恢复出厂设置 -> 烧录后无操作
  - 烧录前 Option Byte 恢复出厂设置 -> 烧录后写入用户自定义 Option Byte
- 支持在线读取目标芯片 Chip ID，方便用户检查 Chip ID，并且对于非连续的 Chip ID 自
- 动按照连续地址的形式给出。
- 支持在线任意地址读取目标芯片内部数据，主要用于开发者 Debug 分析目标芯片，并可任意设置读取地址和读取大小，详见相关章节，主要用于：
  - 读取任意内存数据
  - 读取任意 Flash 数据
  - 读取任意寄存器数据等
- 支持软件自动在线升级（有网络的情况下）
- 支持多国语言，完整支持
- 支持主流 ARM Cortex-M 芯片 (目前支持 STM32 全系列，适配不断更新中...)
- 支持多种擦除方法：按 Page 擦除、全片擦除、Bank 自动识别擦除 (无须用户选择)
- 支持多种电压选择，1.8V/3.3V/5.0V/或者是用户自定义参考电压
- 支持编程速度自由调节，5Khz~10MHZ 速度自由调节
- 支持蜂鸣器在线提示音，定义规则请参考信号定义描述章节
- 支持序列号写入，可设置序列号地址、初值、步长、大小端模式
- 支持序列号 10 进制和 16 进制切换显示
- 支持序列号地址、合法性检查（自动检查是否和其他设置地址重叠）

- 支持离线烧录次数设置，最高可到 42 亿次
- 支持自动芯片检测，芯片放入时可自动启停烧录
- 支持设置自动芯片检测的稳定时间，默认 100ms
- 支持设置自动烧录芯片时拿走延时时间，默认 100ms
- 支持烧录次数设置信息的 10 进制和 16 进制显示切换
- 支持烧录次数设定参数的合法性检查
- 支持烧录完目标芯片后启动目标芯片(Reset & Run)
- 支持烧录完目标芯片后关闭电源输出，在批量烧录时，通过烧录完关闭电源输出，可
- 有效保护芯片不会被带电从烧录座取下
- 支持设置烧录前和烧录后供电稳定时间或者掉电稳定时间（开启关闭电源输出选项有效）
- 支持设置 RESET 引脚信号控制,支持以下三种模式
  - 输出常低电平
  - 关闭输出(高阻状态)
  - 输出复位后关闭(烧录前产生复位信号)
- 支持数据校验：校验烧录到目标芯片的数据是否正确，默认开启
- 支持 UID 创芯工坊官方授权服务器在线芯片授权烧录
- 支持 Power Writer 内置离线授权算法对芯片进行授权
- 支持灵活设置 Power Writer 内置离线授权算法的密钥存放地址，密钥大小(4/8/12byte)，以及密钥的大小端模式，以及用户自定义密码功能
- 支持用户自定义离线授权算法功能，而不是已有的几种模式
- 支持自动生成随机离线授权算法，Power Writer 可自动生成随机离线授权算法，每一种都是唯一的，重复的可能性几乎为零
- 自动生成随机离线授权算法，支持算法强度检查，并提供手动调整参考建议。用户可手工调整，或者重新生成，直到满意为止
- 支持之定义离线授权算法导出 Sample Project，用户只需将导出的源码编译到项目中即可实现离线授权算法，方便快捷
- 通过完善的系统内核调度机制，Power Writer 可同时执行并发操作
- 支持完整的在线操作日志显示，及时提醒用户，日志主要包含以下几种颜色
  - 浅蓝：代表一般性操作结果
  - 绿色：代表操作成功
  - 红色：代表错误，或者其他一些关键信息，比如 Power Writer 断开链接
  - 黄色：代表警告，操作可能有问题，或者是提示设置可能有问题
- 支持日志重置、保存日志、想保存下操作记录，试试将日志保存下来备份、以便将来遇到问题时可以清楚地看到上一次的操作流程
- 支持加密保存，加载烧录配置参数
- 支持完整芯片 Flash 空间映射，用 HEX 视图显示，用户可以直观地看到加载的原始数
- 据
- 支持 Flash 数据区复制，粘贴操作，用户可以对 Flash 数据区进行编辑
- 支持 Flash 数据区地址跳转，用户可以快速跳转到指定的地址
- 支持多段固件烧录功能,并且对固件数量没有进行限制
- 多段固件可实时看到固件的起始地址，结束地址，大小和 CRC32 信息
- 支持自定义固件地址，用户对 Bin 格式固件自定义地址，并对地址合法性进行检查

- 支持丰富的固件格式，bin/Hex/S19/pkg 格式 (pkg 格式为创芯工坊自定义格式)
- 支持添加随机数组功能,并且不限制随机数组的数量（开发中）
- 支持完整的芯片扇区表查看，可以直观地看到芯片的扇区信息，起始地址，结束地址和大小
- 支持自动识别固件对应的扇区信息，选择固件时可以看到固件对应的扇区
- 支持在线自定义扇区擦除，用户可以在线对指定的扇区进行擦除
- 支持 Bank 自动识别，针对单/双 Bank 芯片，用户同步修改、读取 Option Byte 时，扇区表会跟随 Option Bytes 设置进行切换
- 支持随机填充选定的扇区表，用户可以对 Flash 中剩余的空闲部分填充随机数据
- 支持通过 CTRL 控制信号，启动离线烧录
- 支持手动烧录：按下按键进行烧录
- 支持 LED 状态指示：LED 指示脱机下载器运行状态，四个独立的多彩 LED 显示 POWER / BUSY / OK / NG 状态，而不是通过一个 LED 显示，更加直观
- 支持创芯工坊服务端远程下载
- 支持创芯工坊芯片量产授权
- 支持第三方自建授权服务器授权
- 支持机台信号给出，可以给出 NG、OK 信号
- 支持设备当前配置读取 (注：部分敏感信息不会被读取显示)
- 支持固件一键升级：简单方便，确保产品升级方便
- 软件自动设备检测：插入设备到 USB 软件自动检测连接，无须手动操作
- 支持烧录器到目标芯片数据加密，可防止用户固件数据被窃取
- 支持数据加密：设备和 PC 通信数据多次加密，确保数据安全性
- 设备存储数据加密：设备中固件存储数据采用多种加密算法确保数据无法被解密
- 支持烧录地址有效性自动检测，确保程序运行正确
- 多固件同事烧录支持地址重叠检查
- 兼容的 SWJ 引脚信号
- 支持 ARM 内核芯片的 Debugger / Trace 功能，Power Writer 不仅仅是一个在线烧录器和量产烧录器，同时也是一个全功能的 Debugger，不仅支持 Cortex-M 芯片的调试，同时也支持 Cortex-A CPU 的 debug
- 主流的 USB HID 通信方式,用于作为 Debugger 调试 ARM 内核芯片
- 支持读取最后一次离线操作结果，用来快速自查错误
- 支持在线模拟离线烧录,通过此功能,可以在线的方式写入配置的所有信息,参考智能自动编程

2.4 Power Writer 使用前准备

2.4.1 Power Writer 硬件使用前准备

在使用 Power Writer 之前，需要做一些必要的准备：

- 准备好 Power Writer 系列通用客户端软件，可以从创芯工坊官网获取。
  官方网站如下：https://www.icworkshop.com/user/clientDownload
  也可以关注创芯工坊的公众号，获取 Power Writer 通用客户端软件，创芯工坊公众号见首页二维码，以及通过创芯工坊 Power Writer 系列用户交 流群组获取软件的最新版本。

- Power Writer HID 首次连接到系统时，系统会自动安装 HID 驱动，请等待驱动安装完毕，HID 驱动集成在系统中，无须额外提供。

图 2.4.1.1 Power Writer HID 设备显示名称

安装成功之后从设备列表中看到 PowerWriter 的 Debugger 设备(DAPLINK CMSIS-DAP)。

- Power Writer 同时会枚举出一个串口设备，见下图所示,针对 Win8/Win10 等高版本系统，同样可以使用系统自带的串口驱动：

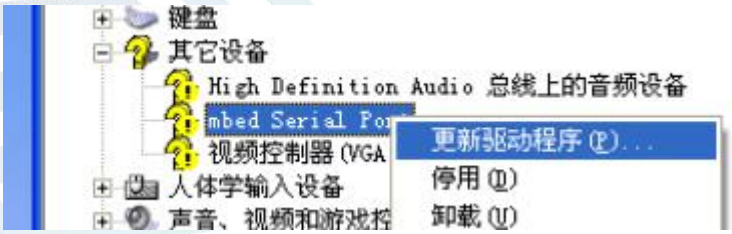


图 2.4.1.2 串口没有安装驱动时显示

- 对于旧版本系统(Windows XP/Windows 7)，由于系统不自带串口驱动，用户需要手动安装驱动，驱动存放在 Power Writer 应用软件的 driver 目录下：Power writer/driver/powerwriter_driver.exe，然后从设备管理器找到端口：mbed Serial Port，右键手动更新驱动程序 (此处采用了 ARM 公司提供的 mbed Serial 驱动，Win10 可采用系统自带驱动)，手动安装 Power Writer 驱动参考如下步骤:
  - ✧ 1：选择 mbed Serial Port -> 鼠标右键更新驱动程序(P),进入到驱动向导，如图 2.4.1.2 所示。如果有跳出从 update 更新，选择否，从系统中选择
  - ✧ 2：选择从列表或者指定位置安装，如图 2.4.1.3 所示，然后进入下一步。
  - ✧ 3：选择不要搜索，我要自己选择要安装的驱动程序，如图 2.4.1.4 所示。然后进入下一步。
  - ✧ 4：从列表中往下找，找到 端口(COM 和 LPT)，如图 2.4.1.5 所示，然后进入到下一步。
  - ✧ 5：选择 mbed,右侧选择 mbed Serial Port 驱动，然后进行下一步安装，如图图 2.4.1.6 所示，

接下来可能会提示驱动未经过数字签名，需要执行继续安装，一步一步往下，即可安装好驱动程序。

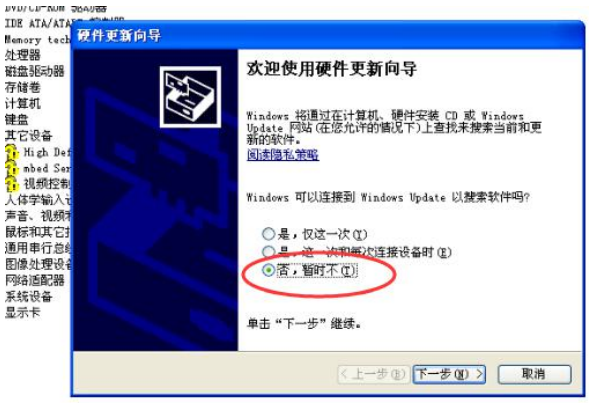


图 2.4.1.3 选择否，没有此步骤则跳过。

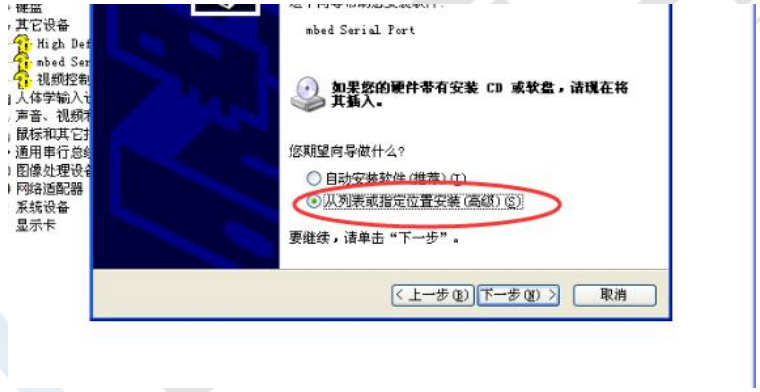


图 2.4.1.4 从列表或指定位置安装

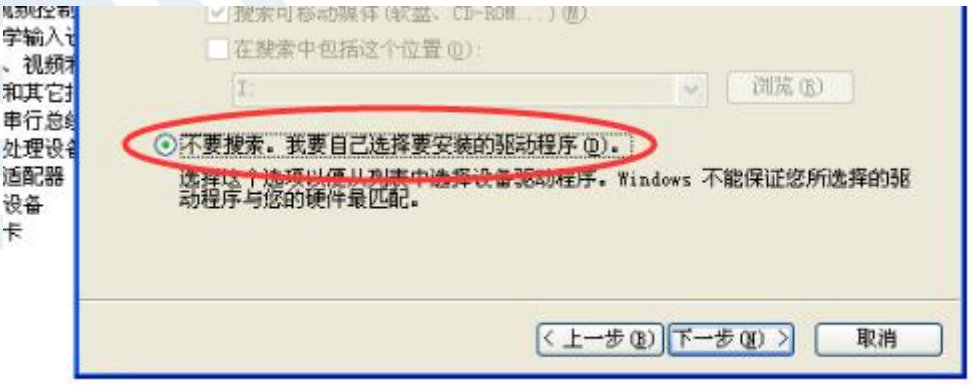


图 2.4.1.5 不要搜索，我要自己选择要安装的驱动程序。

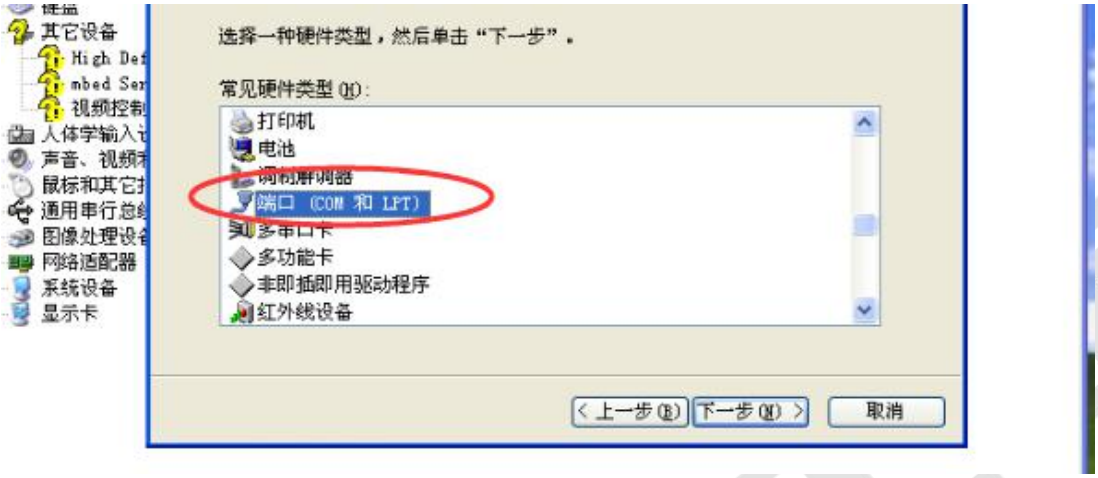


图 2.4.1.5 端口(COM 和 LPT)

图 2.4.1.6 选择厂家为 mbed 型号选择 mbed Serial Port 驱动

- 当驱动安装完成后，采用 Type C USB 线连接电脑与 Power Writer，如下图所示：

图 2.4.1.7 Power Writer 连接目标芯片

- 当驱动和线材都连接好之后，启动 Power Writer 客户端，进入正常的操作流程，见下图所示：

图 2.4.1.8 Power Writer 用户主界面 (Windows xp 系统下)

2.4.2 Power Writer SDK 获取方式

- 如果是利用创芯工坊进行在线量产,同时需要安装 ICWorkShop 客户端,可以从下面的连接下载 https://www.icworkshop.com/user/clientDownload，有关创芯工坊固件远程烧录请参考官方文档教程，链接如下：https://www.icworkshop.com/article/helpCenter/37/33 (创芯工坊帮助中心)

- 对于 ICWKEY 用户，由于需要加入创芯工坊 sissdk 进行二次开发，故需要从创芯工坊官网获取 sissdk 的开发资料包，sdk 的使用方式，以及文档请参考 ICWKEY 用户手册，以及 Demo 项目。

- 对于使用创芯工坊提供的远程授权服务器的用户，如果需要二次开发私有服务器，请从 http://chiplicense.icworkshop.com/admin.php/index/index 获取更多信息
  获取在线授权的在线授权服务器预览图如下：

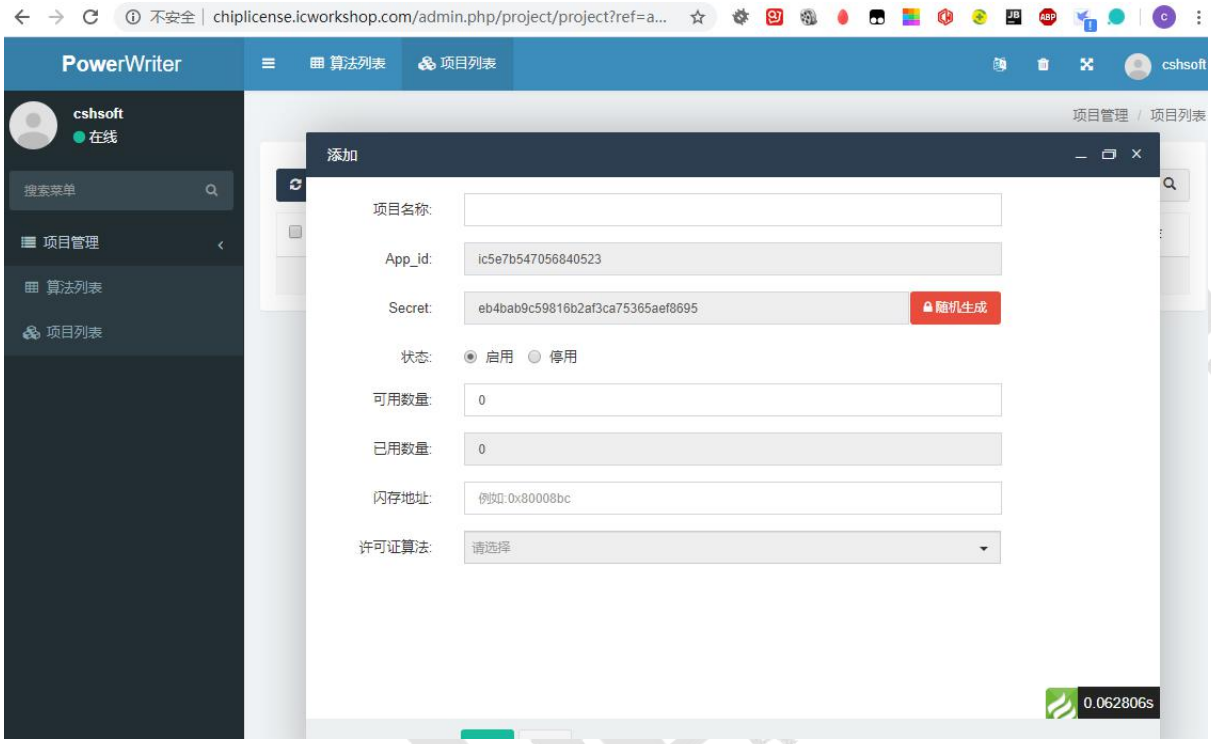


图 2.4.1.6 Power Writer 远程授权后台

2.5 Power Writer 以及配套产品硬件信息

本章节详细介绍了 Power Writer 硬件相关描述，信号定义，接口定义，用户可以通过查阅本章节获取实际使用时的硬件配置信息。

2.5.1 Power Writer 硬件信息

本节主要介绍 Power Writer 硬件有关信息,包括发货包装、端口定义、状态信号，用户可以在这里找到 Power Writer 的端口定义信息

2.5.1.1 Power Writer 包装清单

Power Writer 包装预览如下图所示

图 2.5.1.1 创芯工坊 Power Writer 包装图

发货清单:

- ✧ Power Writer 主机 1 台
- ✧ 创芯工坊定制版端口信号线 1 套
- ✧ Type-C USB 数据线 1 条

2.5.1.2 Power Writer 端口引脚定义

Power Writer 支持 SWJ 和 SWIM 口信号输出，同时在 JTAG 标准定义的基础上扩展了部分信号引脚见下图所示：

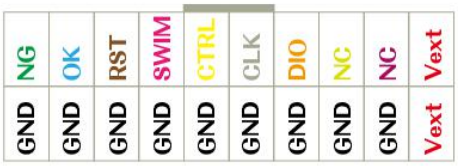


图 2.5.1.1 Power Writer 端口示意图

引脚分类	引脚名称	引脚功能
电源	Vext	电源输入输出(基准可供电)
	GND	电源接地线
SWD/SWJ 接口	DIO	SWD 接口的数据线 SWDIO
	CLK	SWD 接口的时钟线 SWCLK
	RST	SWD 接口的硬件复位线 RESET
SWIM 接口	SWIM	SWIM 接口数据线

机台信号	RST	SWIM 硬件复位线
	CTRL	控制烧录(低有效)
	OK	烧录成功状态输出(高有效)
	NG	烧录失败状态输出(高有效)

注:

①. Vext 线上的电压即信号线的参考电压。可以通过配置软件设置输出 1.8V， 3.3V，5V 电压， 也可以配置为外部参考电压，参考外部的芯片的工作电压。

②. 通过此引脚输入一个 3.3V->0V 的 > 40ms 的低电平将触发一次烧录。

Power Writer 支持从 USB 接口取电和从烧录端口 Vext 取电：

供电方式	输入电压	说明
USB	DC 5V	请确保输入电压足够
Vext	DC 3.3 -5V	低于或高出范围可能无法工作

表 2.5.1.2 Power Writer 供电方式

当从 USB 接口供电时，脱机下载器可以对外输出 1.8V， 3.3V， 5V 三个等级电压，也可以不输出电压，信号线会匹配目标芯片连接到 Vext 线上的电平。

(注: PowerWriter 的供电建议采用 USB 供电，Vext 为供电给目标芯片)

2.5.1.3 Power Writer Type-C USB 端口定义

Power Writer 采用 USB Type-C 接口作为 通信接口

图 2.5.1.2 Power Writer Type-C USB 接口

Type-C USB 端口用于：

- USB 线连接 PC 应用软件用于在线通信，HID/CDC 烧录操作和 Debug
- USB 线用于供电(没有采用扩展供电的情况下)
- 用于连接 ICWKEY 用于生产时的离线授权

(注：如用户使用自己已有的 Type -C 数据线，请留意数据线的负载能力，以及是不是可以用于 USB 正常通讯的线材)

2.5.1.4 Power Writer 主面板定义

图 2.5.1.4 主面板信号定义

主面板信号定义如下：

- POWER：电源指示灯(蓝色)
- STATUS：状态指示灯(黄色)
- NG：烧录失败指示灯(红色)
- OK：烧录成功指示灯(绿色)

主按钮：启动离线烧录 **(**注：当 Power Writer 连接 PC 端软件时，无法启动离线烧录按钮功能**)**

2.5.1.5 Power Writer 信号定义汇总

下表是 Power Writer 的信号定义汇总：

信号名称	信号描述
POWER LED	上电时常亮
STATUS LED	上电无操作时灯不亮
	与应用软件通信时闪烁，闪烁频率跟随通信速度设置
	离线烧录时闪烁，闪烁频率跟随通信速度设置
	当连接到目标芯片时，会常亮
NG LED	当操作有误，读取、擦除、编程等操作失败时灯亮，直到新的操作到来熄灭
OK LED	当读取、擦除、编程等操作成功时灯亮，直到新的操作到来时熄灭

蜂鸣器频率定义	PWM 频率为 2.7K Hz (定义如下)
蜂鸣器次数定义	上电无操作时滴一声
	当连接上目标芯片时滴一声
	当烧录成功或者下载离线档案成功时滴两声
	当操作失败时滴三声
	当离线烧录次数为 0 时，滴四声
按键	当进行离线烧录时有效(松开触发，长于 1S 忽略长按)
OK 信号脚	当操作成功时输出高，有新操作时清 0
NG 信号脚	当操作失败时输出高，有新操作时清 0
CTRL 信号脚	输入>=40ms 的低信号，启动一次离线烧录

2.5.2 ICWKEY 硬件信息 (补充信息)

ICWKEY 是创芯工坊为 Power Writer 配套开发的一款基于 ECDSA 非对称算法授权的硬件模块，本手册提供与 Power Writer 关联的部分 ICWKEY 信息，更详细的使用手册，SDK 开发教程，源码，demo 请参考 《ICWKEY 用户手册 RM0002.PDF》

2.5.2.1 ICWKEY 包装清单

ICWKEY 包装预览如下图所示

图 2.5.1.1 创芯工坊 ICWKEY 包装图(仅供参考,以实际为准)

发货清单:

- ICWKEY 主机 1 台
- Type C USB 供电线一条 1 条

2.5.1.2 ICWKEY Type C USB 端口定义

ICWKEY 采用双端 USB Type C 接口，一端用于供电，另一端用于和 Power Writer 通信，见下图所示：

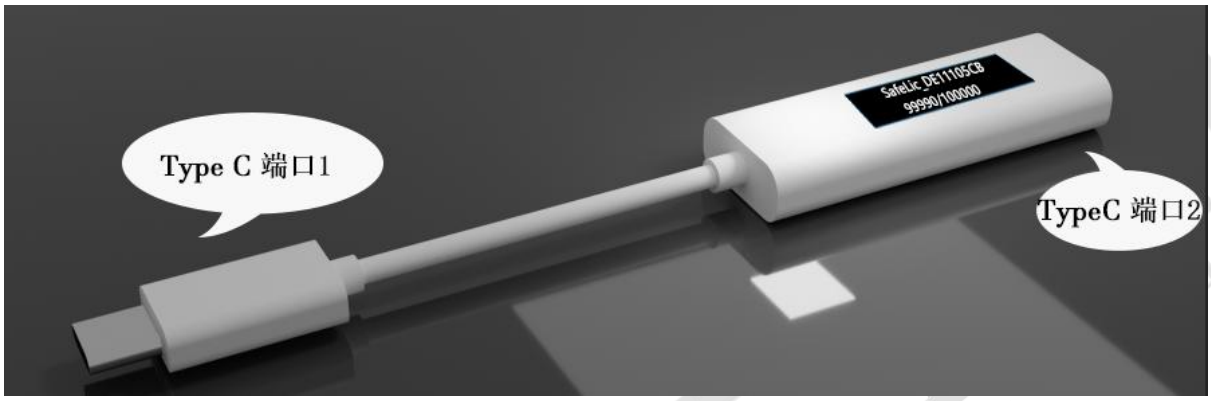


图 2.5.1.2 Power Writer Type C USB 接口

Type C USB 端口用于：

- Type C 端口 1：用于和 Power Writer 连接通信，并同时供电给 Power Writer
- Type C 端口 2：用于给 ICWKEY 和 Power Writer 整体供电，以及用于 ICWKEY 配置软件对 ICWKEY 进行授权配置

2.5.1.4 ICWKEY 主面板定义

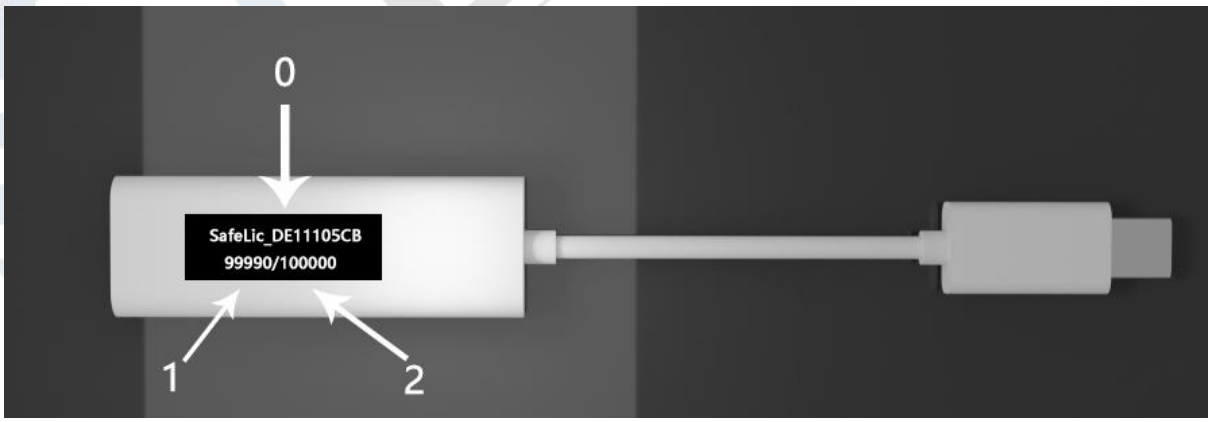


图 2.5.1.4 主面板信号定义

主面板定义如下：

- 0：显示项目名称(Project 代码,系统生成的默认格式为: SafeLic_XXXXXXXX)
- 1：显示当前项目剩余的授权次数
- 2：显示当前项目总得授权次数

3 ： Power Writer 软件系统

本章节将详细地介绍 Power Writer 相关的软件系统，用户可以作为软件的使用指南，或者是参考手册，当遇到不理解的内容时，可以随时查阅，本章节内容较多，用户可以选择从头到尾详细的阅读一遍，以便对 Power Writer 以及创芯工坊系统有一个系统地理解，或者是查阅自己想了解的部分，或者直接跳到第 4 章的应用指南，本章节主要包含以下几个章节的内容：

- Power Writer 应用软件详解
- ICWKEY 应用软件概览
- ICWKEY SDK 开发指引
- 创芯工坊后台管理系统概览
- 创芯工坊 License Server 概览

(注：本章节重点突出 Power Writer 关联的部分，对于 ICWKEY、创芯工坊后台管理系统、License Server、ICWKEY SDK 部分介绍使用流程，以便用户可以从本文档即可快速入手，更详细的资料需要参考对应的使用手册)

ICWKEY 的使用参考：《ICWKEY 用户手册 RM0002.pdf》
ICWKEY SDK 开发参考：《ICWKEY SDK 开发指南 RM0003.pdf》
创芯工坊后台管理系统参考：
https://www.icworkshop.com/article/helpCenter/44/36
创芯工坊 License Server 参考：
http://chiplicense.icworkshop.com/admin.php/index/index

3.1 Power Writer 应用软件详解

本节将详细介绍 Power Writer 应用软件的详细使用方法，用户可以作为参考手册随时来查阅其中的功能，遇到问题时也可以作为工具书来查阅，Power Writer 应用软件的启动界面如下图 3.1.1 所示。

图 3.1.1 Power Writer 应用软件主界面

Power Writer 应用软件主要分为：

- 标题栏
- 菜单栏
- 工具栏
- 选项卡(烧录器设置、选项字节、Program Memory、OTP、EEPROM) (注:OTP、EEPROM 待升级)
- 状态栏

以下将分章节详细介绍各部分的用法。

3.1.1 Power Writer 标题栏

Power Writer 标题栏包含

- Power Writer 版本号
- Power Writer Build 日期时间信息
- Power Writer 最小化，最大化，退出

3.1.2 Power Writer 菜单

3.1.2.1“文件”菜单

文件菜单包含保存项目、项目另存为、加载项目、退出四个常用功能

图 3.1.1.2.1 Power Writer 文件菜单

3.1.1.2. 保存项目

当用户完成项目设置之后，可以将整个项目打包成一个加密的项目文件

- 用户第一次点击文件菜单将会弹出首次保存的路径和设置用户密码，见图 3.1.1.2.2 所示：
  - 密码设置：用户设置项目文件的密码，要求用户输入 16 个字符的密码,注意密码不能输入太短，否则会提示 密码设置错误，如果不想将密码设置成 16 位，可以用一些固定字符代替。
  - 文件路径：密码设置完成之后，需要选择保存文件的路径，点击

文件路径 路径按钮，在弹出的对话框中选择保存的路径，并设置好保存文件名称，然后点击设置完成按钮 设置完成 ，

图 3.1.1.2.2 保存项目文件设置

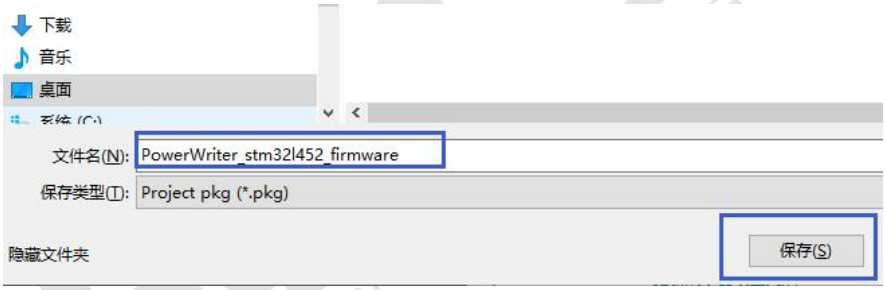


图 3.1.1.2.3 保存文件对话框

保存项目会在状态栏显示项目文件的路径,见图 3.1.1.2.4 所示,同时日 志栏将显示保存的结果03/26-11:53:51:746> 保存成功 ，如果保存失败将显示失败的提示信息。

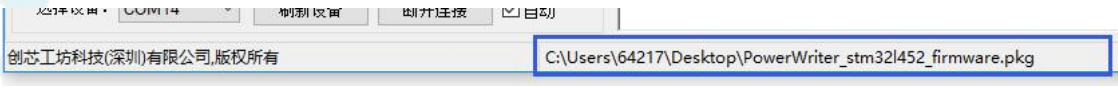


图 3.1.1.2.4 保存项目之后会在状态栏显示保存项目的路径

附加说明：

1：创芯工坊加密文档格式为自研算法，同时具有高强度和良好的性能

2：pkg 项目文档内部不保存任何密码的信息，采用内部自验证机制，如果用户忘记了密码，pkg 项目文件将无法解包，创芯工坊官方也只能通过枚举的方法来尝试还原项目文件,所以请用户牢记自己设定的密码。

3：为何设置为强制 16 位长度，经过我们的 Benchmark 测试加密算法性能，过短的密码在 Hacker 通过极限手段获取到了加密算法原理，可采用暴力测试可能会获取到真实数据，而设置成 16 位暴力破解的时间成本将会非常的高，更多安全特性请参考安全特性章节

■ 第二保存将不会再弹出保存文件对话框，直接保存项目到上次设置的信息

3.1.1.2.2 项目另存为

项目另存为的定义和保存项目的定义区别在于：另存为每次都会弹出新的保存项目信息，而不是保存为上次设置的路径和密码，同其他软件定义的另存为功能上完全一致。

3.1.1.2.3 加载项目

通过加载项目功能，可以加载之前保存的项目文件到 Power Writer 软件中，操作方式和保存项目完全一致：

- 填写项目的密码，密码错误将无法加载。
- 设置项目的路径

3.1.1.2.4 退出

退出 Power Writer 软件，功能同系统标题栏的  。

注：系统在操作过程中将执行退出

3.1.2.2“执行”菜单(功能不断升级中)

执行菜单包含芯片在线操作的常用功能,见图 3.1.3.1 所示

图 3.1.3.1 执行菜单功能

3.1.2.2.1 保存并离线加载

当用户将项目的所有设置完成之后，可以将项目配置到 Power Writer 硬件，用于实际的产品生成，此功能会同时执行 **保存项目** 并加载项目到 Power Writer 硬件，如果操作成功将会提示

，如图 3.1.3.1 所示：

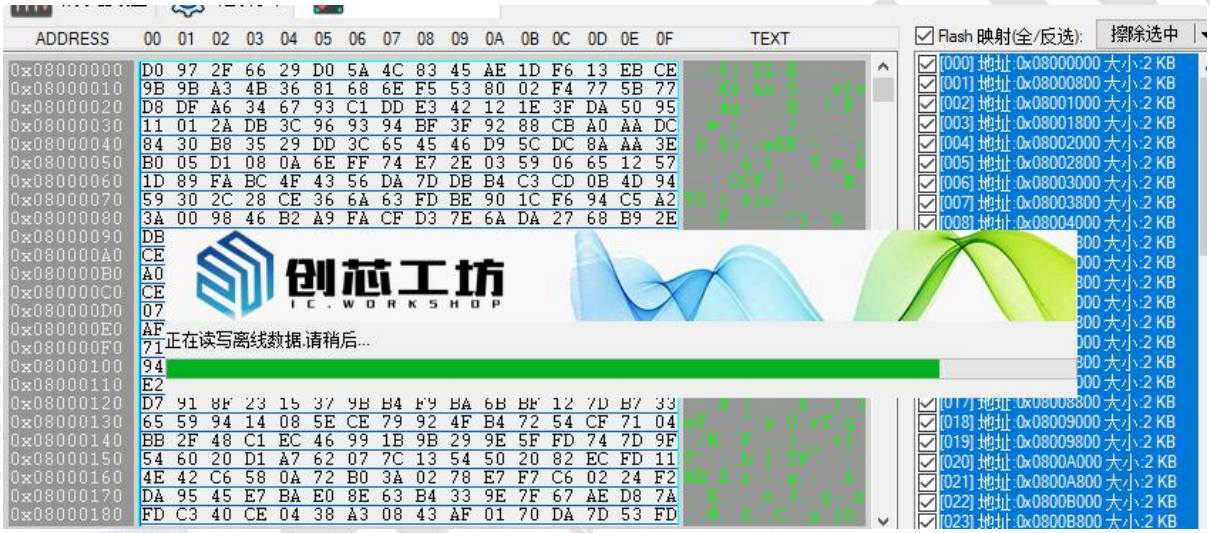


图 3.1.3.1 保存并离线加载演示

3.1.2.2.2 离线读取并保存

如果用户需要将 Power Writer 中已存储的 Project 文件读取回来，可以执行离线读取并保存，此动作将会将项目文件从 Power Writer 读取回，并尝试加载为当前项目，需要用户填写 Project 的密码，如图 3.1.3.2 所示:

注：如果用户密码错误，将无法加载到当前项目，提示 

图 3.1.3.3 离线读取并保存设置

3.1.2.2.3 读取 Program Memory

用户通过读取 Program Memory 功能读取芯片的 Program Memory 数据到 Power Writer 软件，设置如下图 3.1.3.3.1 所示，读取完成后，数据将会在 Program Memory 选项卡显示，如图图 3.1.3.3.2 所示，如果读取失败，则会显示错误信息

- 读取地址：用户可以自由设置读取地址，默认设置为芯片 Program Memory 的首地址
- 读取大小：用户可以设置为 1KB、2KB、4KB、8KB、16KB、32KB、64KB、128KB、256KB、512KB、1MB、2MB、4MB。等
- 整片读取，如果用户需要读取整片的 Program Memory 数据，直接勾选整片读取，默认不勾选

图 3.1.3.3.1 读取 Program Memory 设置

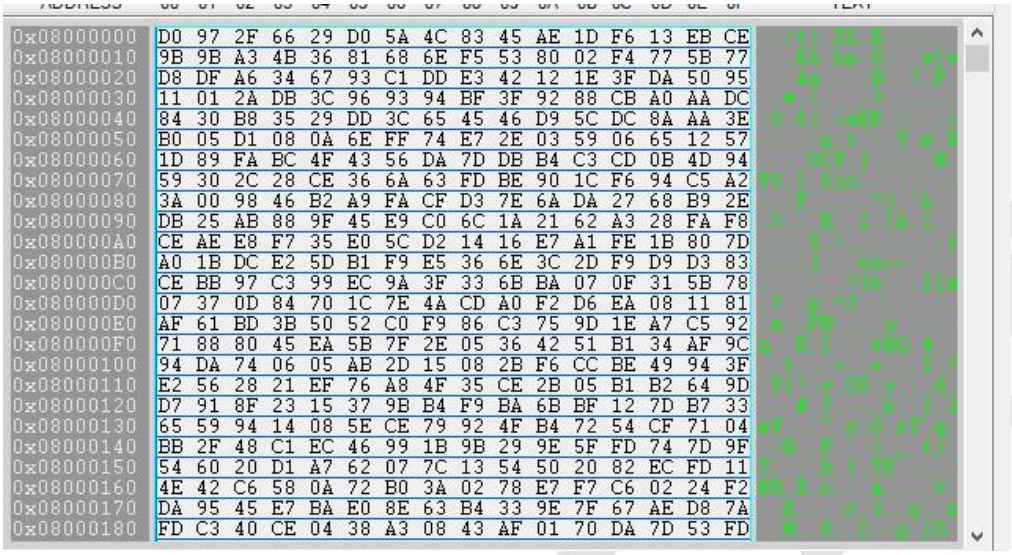


图 3.1.3.3.2 Program Memory 实时显示读取到的数据并高亮

注：1：如果芯片有读保护，则无法读取数据。

2：如果设置参数超过了芯片的容量，则会提示读取失败，后段数据无法读取，一般直接勾选整片读取即可

3.1.2.2.4 查空 Program Memory

用户通过此功能，可以快速检查芯片是不是空片，操作演示如下，测试芯片为 F071CB，测试之前写入了数据，执行查空操作提示如图 3.1.3.4.1 所示：

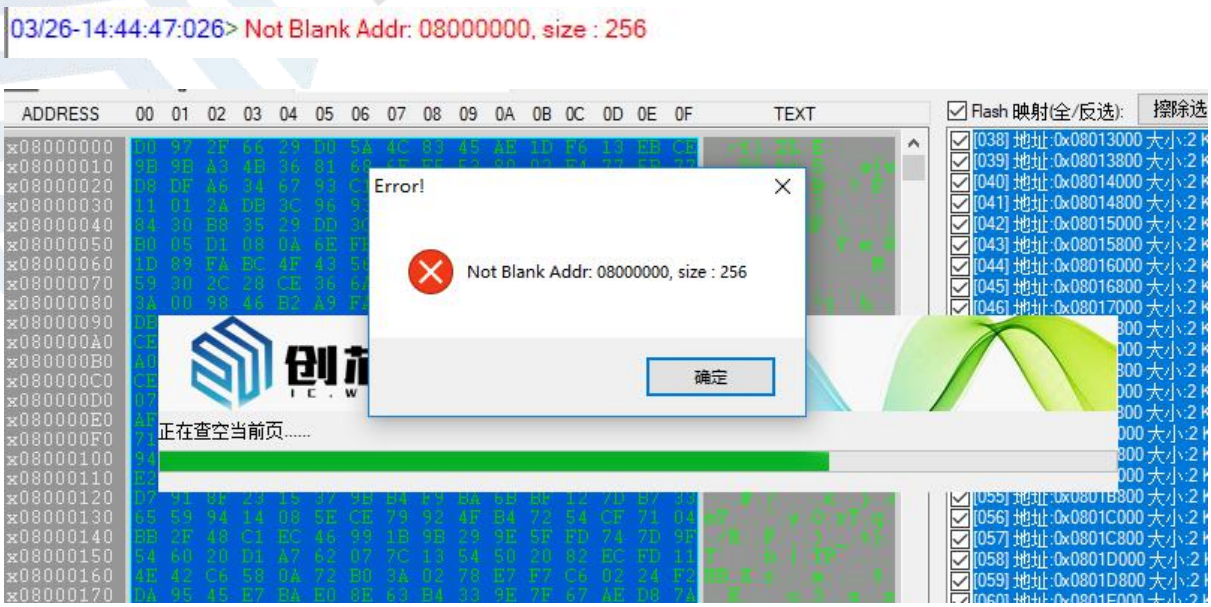


图 3.1.3.4.1 查询芯片是不是空片提示信息

如果芯片为空片，则会在日志栏显示,查空成功,如图 3.1.3.4.2 所示，此演示测试步骤为：

✓ 擦除目标芯片，提示擦除成功
✓ 执行查空操作，提示查空成功

图 3.1.3.4.2 查询芯片是不是空片成功提示

3.1.2.2.5 擦除 Program Memory

用户通过 Program Memory 功能可以快速擦除芯片，此功能执行的是整片擦除，如果需要 Sector 擦除，请参考 Program Memory 的扇区分页位置的扇区擦除功能，擦除成功示范如图 3.1.3.5.1 所示：

图 3.1.3.5.1 擦除芯片成功演示

当芯片有读/写保护时，执行擦除操作会失败，如图 3.1.3.5.2 所示

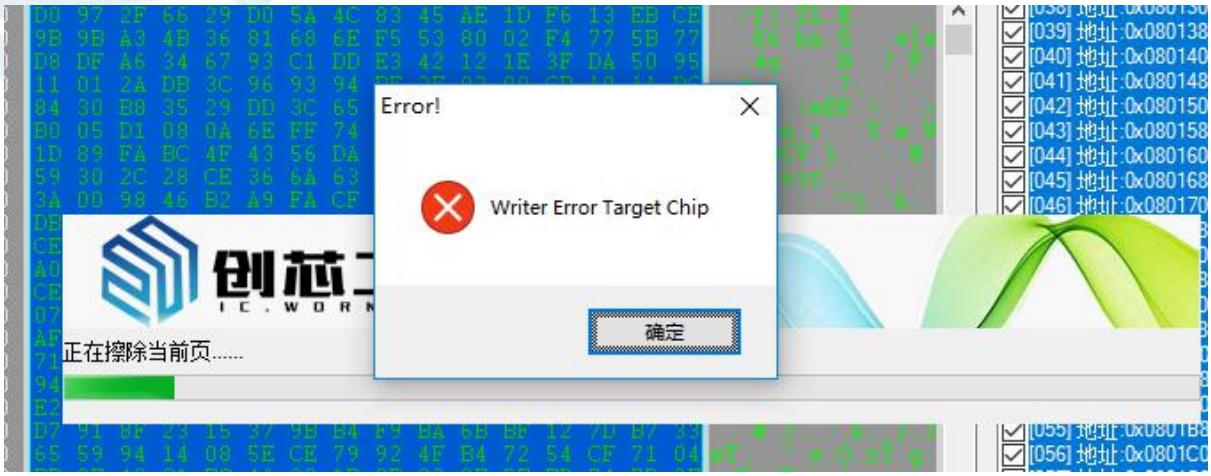


图 3.1.3.5.2 擦除芯片失败

注，如果擦除失败，请按下面步骤进行排查：

- 执行任意一项操作，或者检查 Power Writer 的状态指示灯是不是长亮的，如果是常亮，说明芯片时在线的，如果是灭的说明线路连接有问题
- 执行读取选项字节，检查选项字节中的保护设置是否为 0 级，如果不是，则需要修改成 0 级，然后写入。
- 检查扇区写保护，如果写保护是开启的，去掉，更新选项字节到目标芯片。
- 当选项字节更新之后，再执行擦除即可正常擦除芯片

如果以上步骤都不能擦除芯片，请联系我们的技术支持团队（cs@icworkshop.com）。

3.1.2.2.6 编程 Program Memory

开发人员可以通过 Power Writer 在线对目标芯片进行烧写，烧写时，有以下两点特性需要留意

- 当添加了分段时：执行 Program Memory 操作，写入的是所有的分段固件。分段数量无限制，如图 3.1.3.6.1 所示：

图 3.1.3.6.1 当使用分段烧录功能时烧录的是分段固件

- 当没有添加分段时：执行 Program Memory 操作，写入的是 Program Memory 所有数据。如图 3.1.3.6.2 所示：

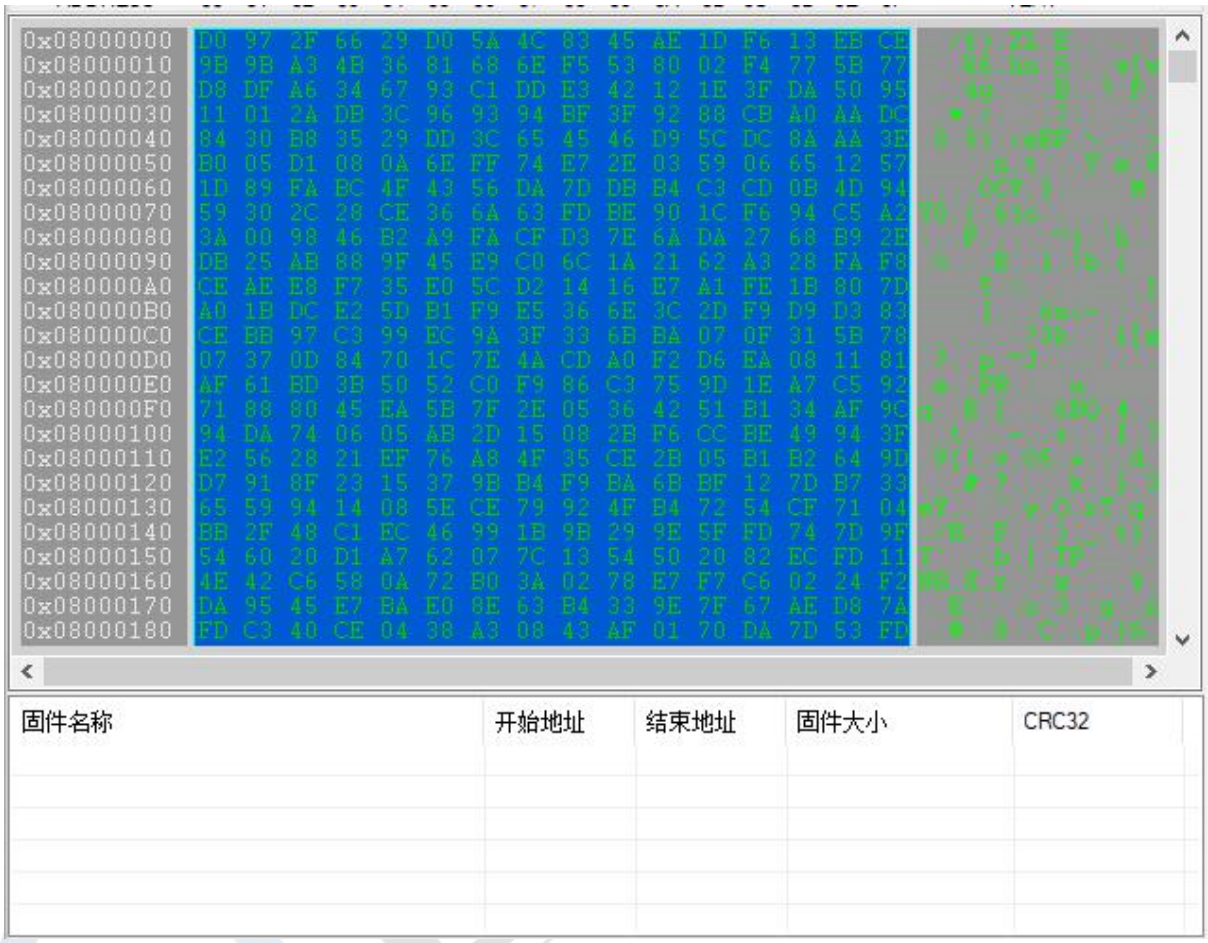


图 3.1.3.6.2 当无分段烧录功能时烧录的是 Program Memory 所有数据

执行 Program Memory 操作成功时，将看到图 3.1.3.6.3 所示的提示信息

图 3.1.3.6.3 执行 Program Memory 操作成功提示

执行 Program Memory 操作失败时，将看到图 3.1.3.6.4 所示的提示信息，并提示写入失败的地址空间。

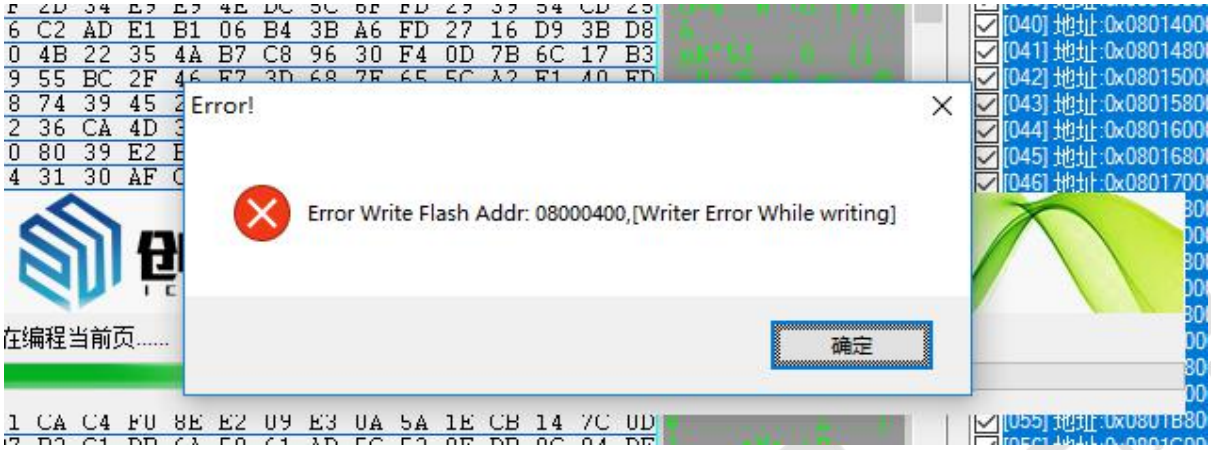


图 3.1.3.6.4 执行 Program Memory 操作失败提示

注：写入失败原因汇总

- Program 区域为非空，处理方法为先擦除芯片，或者擦除制定的扇区
- 芯片保护级别不为 0，处理方法为将芯片保护级别去除，具体选项字节设置
- 芯片的扇区有写保护，处理方法为将扇区写保护去除

3.1.2.2.7 校验 Program Memory

用户可以使用校验功能，对比芯片中的数据和 Program Memory 区域的数据是否一致，需要留意的是，校验功能同样是根据是否分段来做自动处理，如果添加了分段固件，校验分段固件本身的数据，如果没有添加分段固件，校验的是整个 Program Memory 数据区域。校验成功时，将会看到如图 3.1.3.7.1 的所示信息。

图 3.1.3.7.1 校验成功时提示信息

当校验失败时，将看到如图图 3.1.3.7.2 所示信息，从错误信息中可以看到校验的块的首地址，当前校验的数据长度，默认为每次校验 256 个字节。

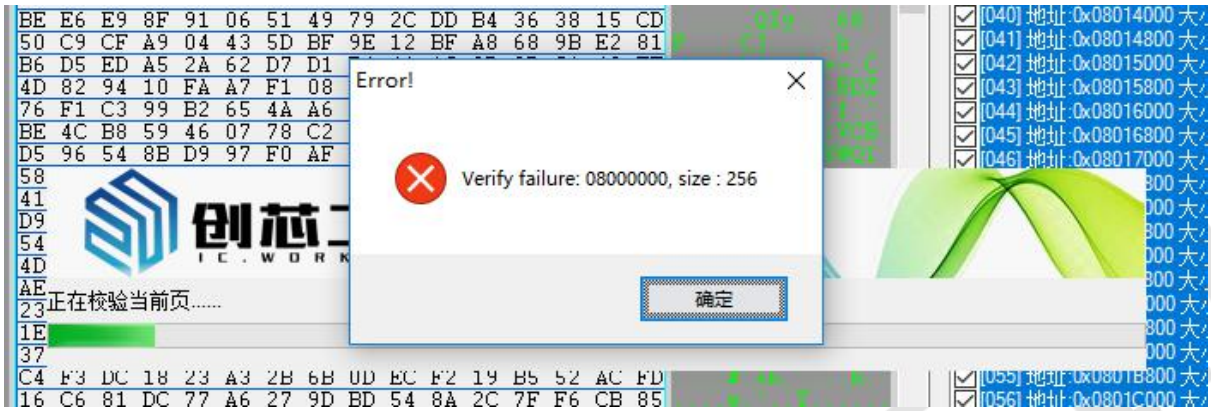


图 3.1.3.7.2 校验失败时提示信息

3.1.2.2.8 Program Memory 自动编程

Program Memory 自动编程功能是 擦除、写入、校验的组合功能，执行此功能时，将会自动执行 Flash 擦除、写入、校验。操作成功时如图 3.1.3.8.1 所示，如果任何一步失败都将提示对应的是失败信息。

图 3.1.3.8.1 自动编程成功时提示信息

3.1.2.2.9 全功能自动编程

全功能自动编程的作用是将用户所有设定的数据，设置项，以在线的方式一次性写入，包括 SN，和 Matrix 绑定数据(授权方式选择 PowerWriter 内置的前提下)，并会同步更新 SN 的起始序号。相当于用在线的方式执行了一次离线烧录(不含 ICWKEY 的方式)，如图 3.1.2.2.9-1 所示

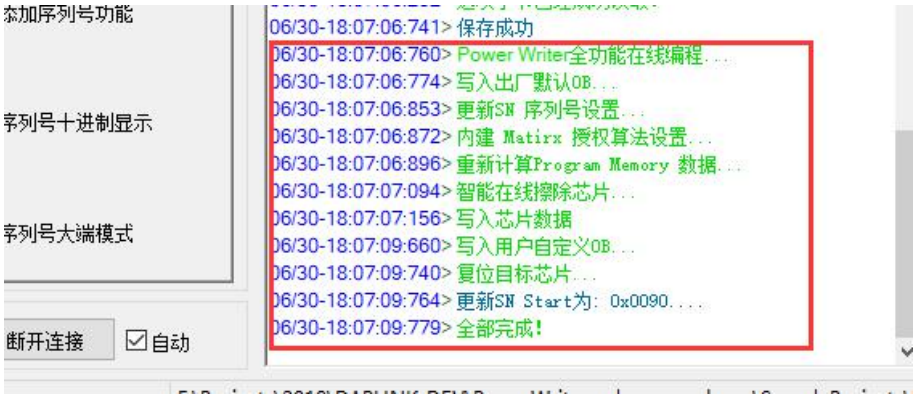


图 3.1.2.2.9-1 智能自动编程成功时提示信息

3.1.2.2.10 复位目标芯片

复位目标芯片功能，可以执行一次目标芯片的复位动作。此操作将同时触发：

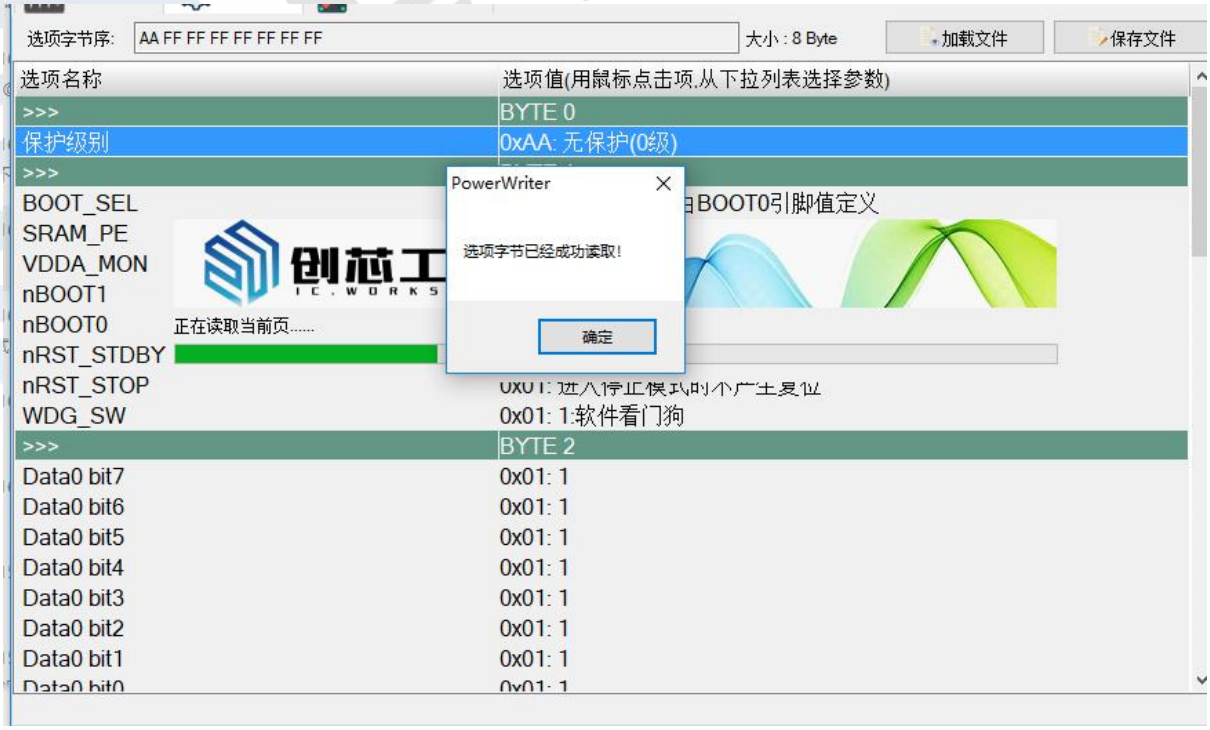
- 执行一次软复位动作。
- RST 引脚输出一次外部复位信号

操作成功将得到成功提示信息，如图 3.1.3.10.1 所示：

图 3.1.3.10.1 复位成功提示信息

3.1.2.2.11 读取选项字节

用户可以使用读取选项字节功能，读取目标芯片的所有选项字节，如图：3.1.3.11.1 所示



图：3.1.3.11.1 读取选项字节成功提示

如果用户选择的芯片和目标芯片不匹配，则无法读取选项字节，提示信息如下图 3.1.3.11.2 所

示：

图：3.1.3.11.2 读取选项字节失败提示

3.1.2.2.12 写入选项字节

当用户需要更新芯片的选项字节时，可以通过此功能，写入自定义的选项字节，操作成功如下图 3.1.3.12.1 所示，选项字节跟官方发布的数据手册保持一致。

图 3.1.3.12.1 选项字节写入成功

注：选项字节写入后，会立即生效，无须用户将芯片断电，Power Writer 内部已经做了自动处理。

3.1.2.2.13 读取 CID

读取 CID 功能可以读取芯片的 CID，连接上芯片之后直接读取即可在日志栏查看芯片的 CID，如图 3.1.3.13.1 所示

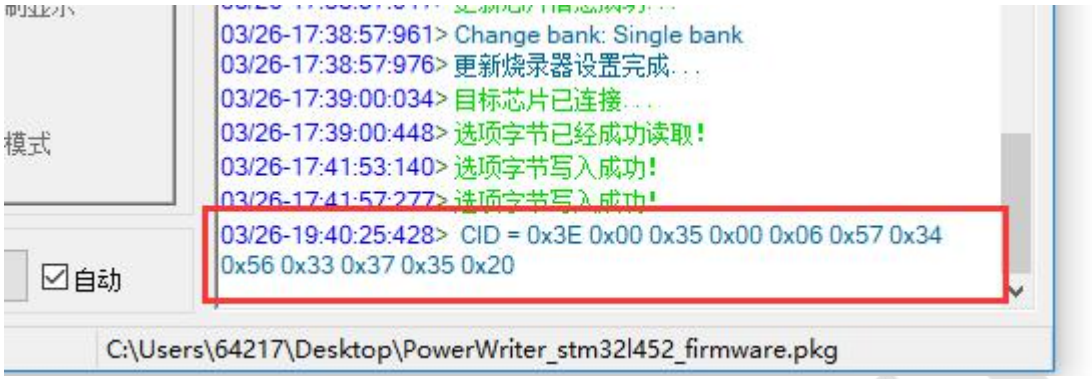


图 3.1.3.13.1 读取芯片 CID

注：部分芯片的 CID 是非连续的，Power Writer 显示的 CID 是将不连续的 CID 连续显示，在 Power Writer 内部自动做了拼接处理。

3.1.2.2.14 任意地址读数据

任意地址读数据是一个非常强大且实用的功能，开发者可以指定任意的芯片内部存储空间地址，包括但是不限于：

- 读取芯片 RAM 数据并显示
- 读取芯片 Flash 数据并显示
- 读取芯片所有寄存器数据并显示

如：读取芯片 RAM 数据，从地址 0x20000000 读取 1K 的 RAM 数据并显示，结果如下：

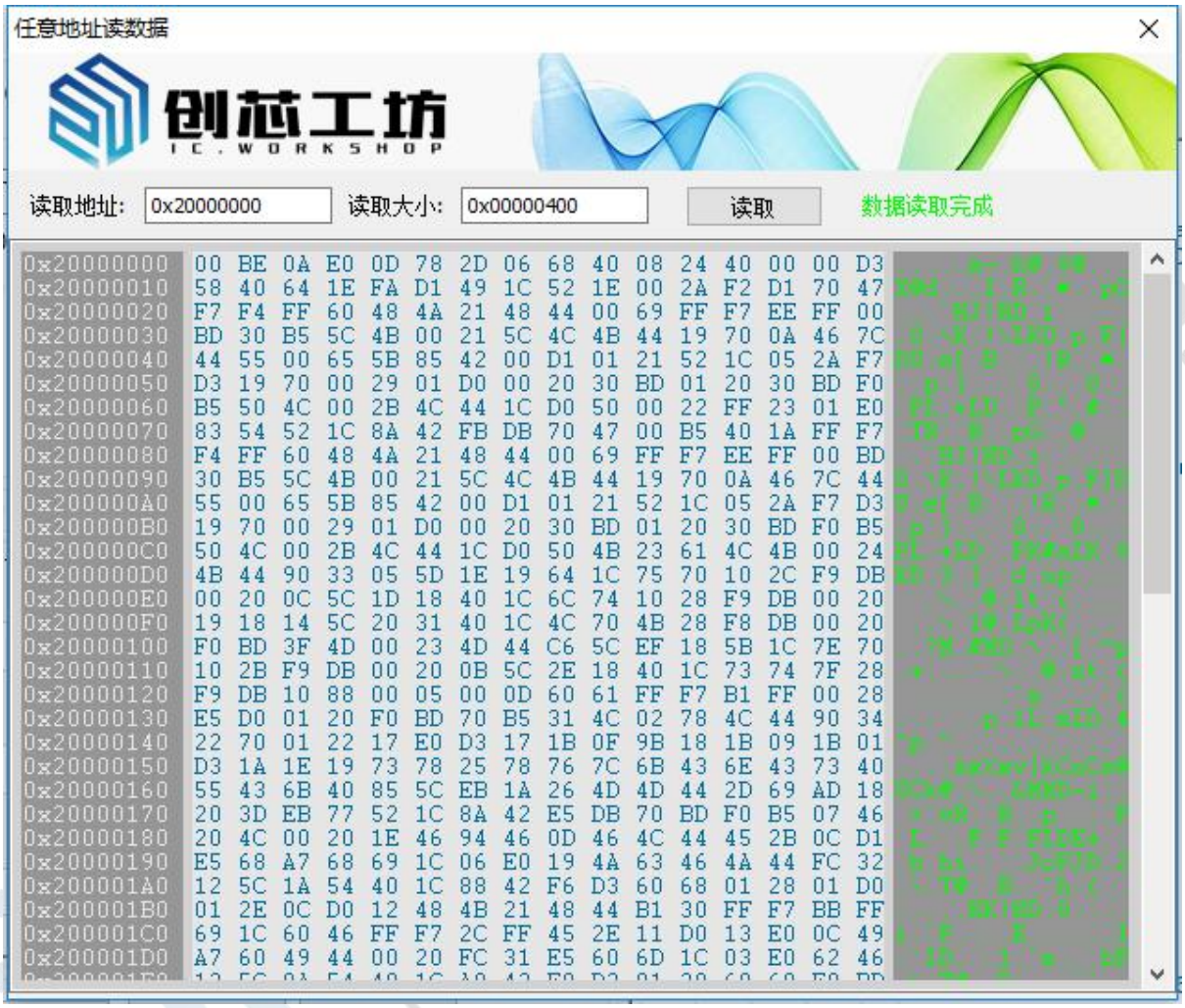


图 3.1.3.14 读取 RAM 数据示范

也可以读取任意的 Flash 地址数据，如读取 0x08000000 1K 大小的数据，结果显示为图 3.1.3.14.2 所示：

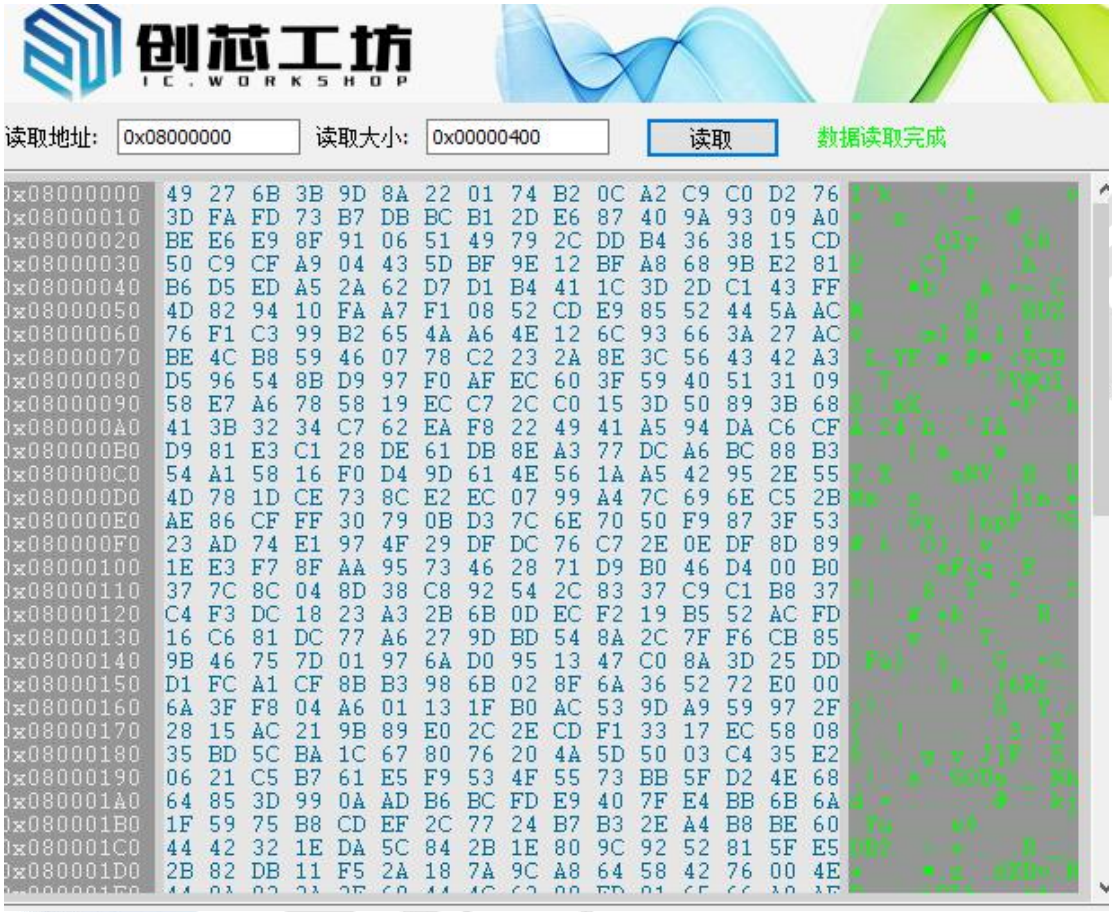


图 3.1.3.14.2 读取 Flash 数据示范

也可以读取芯片内部的 CODE 数据，结果显示为图 3.1.3.14.3 所示：

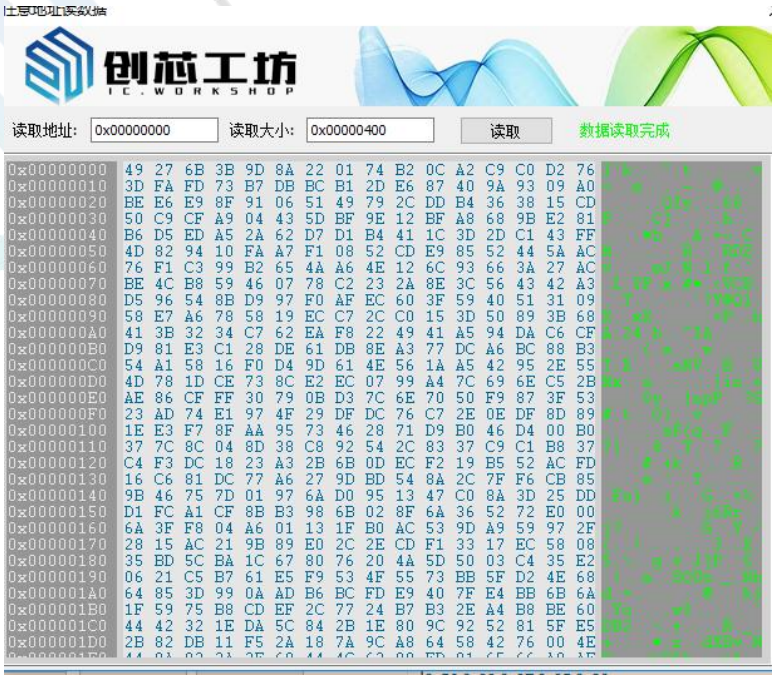


图 3.1.3.14.3 读取芯片内部 CODE 空间数据示范

注：任意地址读数据依赖于你对当前使用芯片的熟练程度，读取芯片内部的数据时，需要设定好需要读取的虚拟地址，以及读取的大小；
读数据可能因为权限问题，或者是由于跨模块时读取数据会导致失败；
此功能跟 ST Programmer 的任意地址读数据功能一致。

3.1.2.2.15 读取最后一次离线操作结果(GetLastOfflineError)

任如果用户在使用 PowerWriter 进行离线生产时，遇到错误，针对 PowerWriter 没有带屏的产品，会无法直观地看到错误的原因，通过此功能，可以直观地查看离线烧录错误的原因。

例如，在没有连接芯片的情况下，启动一次离线烧录。再通过读取最后一次离线操作结果，可以看到如下的提示信息，提示目标芯片错误，如图 3.1.2.2.15.1 所示。

图 3.1.2.2.15.1 读取最后一次离线操作结果演示

3.1.2.3 “帮助”菜单

帮助菜单显示用户的常用帮助信息，比如软件手动升级，跳转创芯工坊官方网站，查看用户手册，以及查看发布信息

3.1.2.3.1 安装驱动

如果找不到驱动程序，Power Writer 无法连接，可通过点击菜单->帮助->驱动安装来安装驱动程序，见图 3.1.2.3.1-1 所示。

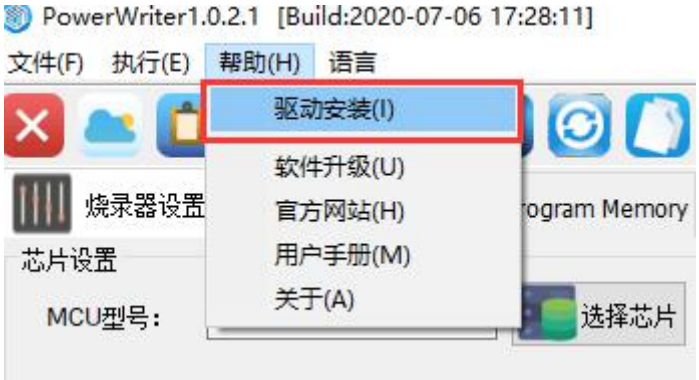


图 3.1.2.3.1-1 驱动安装菜单

如果想要强制重新安装驱动，可以通过找到 PowerWriter 软件安装目录，进入到 driver 目录，找到 powerwriter_driver.exe 进行手动安装，见下图 3.1.2.3.1-2 所示。

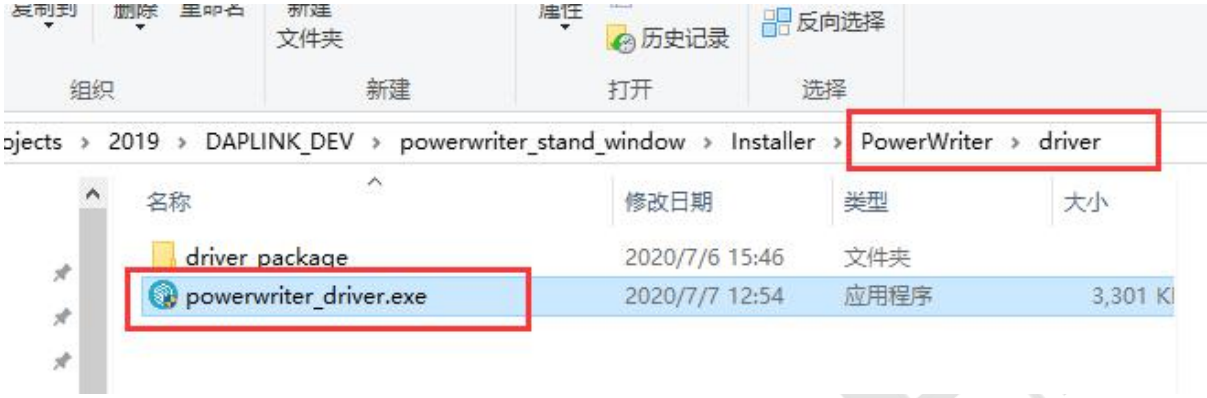


图 3.1.2.3.1-2 自助安装驱动目录

双击运行，将进入驱动安装向导，点击 Install Driver 按钮，安装向导将会对系统驱动进行检测，如果已经安装过，则会提示系统已经安装了驱动，是否覆盖安装，如果需要覆盖安装，点击是，不覆盖安装点击否，如果系统没有安装驱动，则会自动安装，安装成功将会提示成功信息，安装失败将会给出失败的错误原因,见图 3.1.2.3.1-3 所示。

图 3.1.2.3.1-3 驱动安装向导

3.1.2.3.2 软件升级

软件升级是自动提示的，如果软件检测到新版本，则会提示升级。但如果用户不小心关闭了自动更新窗口，又不想重启 Power Writer 软件，则可以点击手动升级，根据系统提示是否对 Power Writer 应用软件进行升级，升级提示界面显示如下：

图 3.1.4.1 软件升级提示

3.1.2.3.3 官方网站

通过点击菜单可以快速达到创芯工坊官方站点：https://www.icworkshop.com

3.1.2.3.4 用户手册

点击之后用户可以查看和 Power Writer 一起发布的最新用户手册，《Power Writer 系列用户参考手册 RM001.pdf》中的内容，也就是本手册中的内容

3.1.2.3.5 关于

关于菜单可以查看 Power Writer 应用软件的发布信息，以及创芯工坊的用户协议，如下图所示

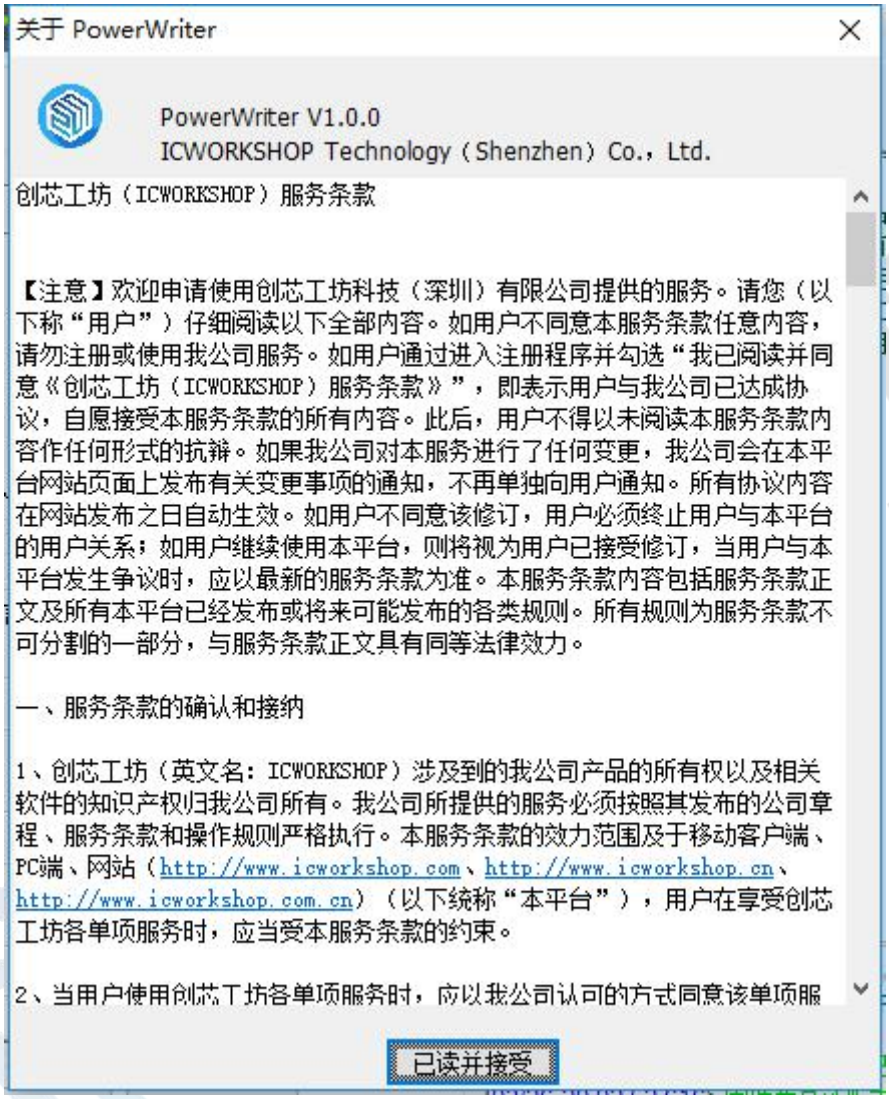


图 3.1.4.1 创芯工坊 Power Writer 发布信息以及用户协议

3.1.2.4“语言”菜单

3.1.2.4.1 中文

软件默认会根据系统设定选择中文还是英文界面，用户可以通过菜单切换到英文界面，下图显示的是中文界面的效果

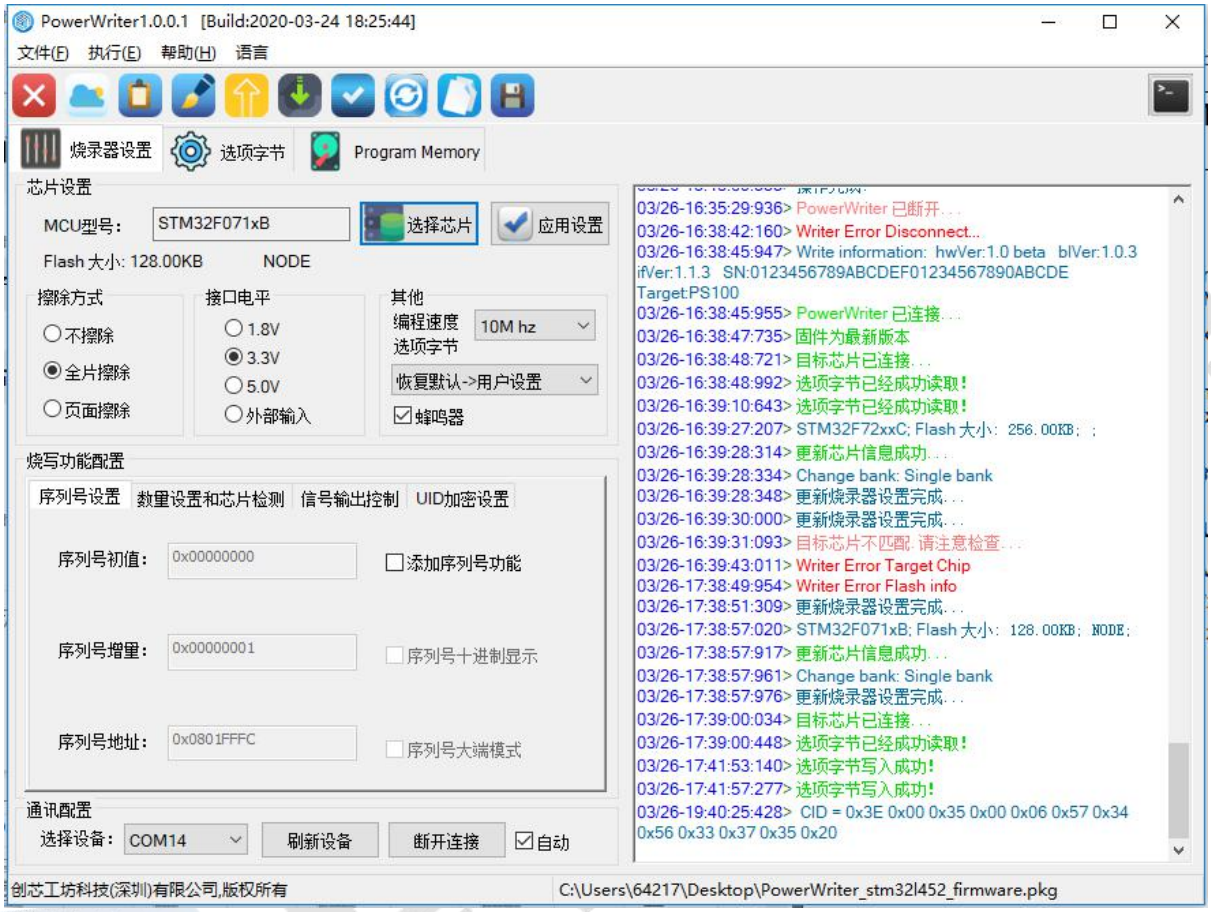


图 3.1.5.1 Power Writer 中文界面显示效过

3.1.2.4.2 英文

软件默认会根据系统设定选择中文还是英文界面，用户可以通过菜单切换到英文界面，下图显示的是英文界面的效果

图 3.1.5.1 Power Writer 英文界面显示效果

3.1.3 Power Writer 工具栏

本章节将重点介绍工具栏的功能，工具栏作为常用功能的快捷入口设立，如图 3.1.3 所示，从左到右分别包含如下功能，

- 退出
- 固件升级
- 当前 TAB 查空
- 当前 TAB 擦除
- 当前 TAB 读取
- 当前 TAB 写入
- 当前 TAB 校验
- 自动编程
- 加载文件到 TAB
- 保存当前 TAB 到文件

图 3.1.3 工具栏

3.1.3.1“退出”按键

工具栏的退出按钮跟文件菜单中的退出按钮，窗口右上角的 X 按钮是同样的功能，不做额外的说明

3.1.3.2“固件升级”按键

固件升级按钮提供手动检查当前烧录的固件是不是最新版本，如果固件为最新，在日志栏将看到 

的提示，如果有新的固件，则会提示更新固件提示。

注：固件升级是自动操作的，当烧录器每次连接到 PowerWriter 时，会自动检查固件是不是最新版固件，无须用户手动点击升级按钮

3.1.3.3“当前 TAB 页查空”按键

当前 TAB 页查空功能支持查空 当前 TAB 页面的数据，详细的支持特性如图表 3.1.3.3 所示

TAB 名称	是否支持
烧录器设置 TAB	
选项字节 TAB	
Program Memory TAB	✓ 支持，等同于查空 Program Memory 菜单
OTP Memory TAB (开发中)	✓ 支持
EEPROM TAB (开发中)	✓ 支持

表 3.1.3.3 当前 TAB 页查空支持的 TAB

注：“开发中”的功能表示仍然在开发当中，暂时还无法使用。

3.1.3.4“当前 TAB 页擦除”按键

当前 TAB 页擦除功能支持擦除当前 TAB 页面的数据，详细的支持特性如图表 3.1.3.4 所示

TAB 名称	是否支持
烧录器设置 TAB	
选项字节 TAB	
Program Memory TAB	✓ 支持，等同于擦除 Program Memory 菜单
OTP Memory TAB(开发中)	✓ 支持
EEPROM TAB(开发中)	✓ 支持

表 3.1.3.4 当前 TAB 页擦除支持的 TAB

注：“开发中”的功能表示仍然在开发当中，暂时还无法使用。

3.1.3.5“当前 TAB 页读取”按键

当前 TAB 页写入功能支持写入当前 TAB 页面的数据，详细的支持特性如图表 3.1.3.5 所示

TAB 名称	是否支持
烧录器设置 TAB	✓ 支持
选项字节 TAB	✓ 支持，等同于读取选项字节
Program Memory TAB	✓ 支持，等同于读取 Program Memory 菜单
OTP Memory TAB (开发中)	✓ 支持
EEPROM TAB (开发中)	✓ 支持

表 3.1.3.5 当前 TAB 页读取支持的 TAB

注：“开发中”的功能表示仍然在开发当中，暂时还无法使用；
烧录器设置 TAB 页面读取，将会读取烧录器的设置到 Power Writer 软件，但是部分敏感信息将不会同步到 Power Writer，如图 3.1.3.5.2 所示。

图 3.1.3.5.2 读取烧录器设置页敏感数据提示

3.1.3.6“当前 TAB 页写入”按键

当前 TAB 页写入功能支持写入当前 TAB 页面的数据，详细的支持特性如图表 3.1.3.6 所示

TAB 名称	是否支持
烧录器设置 TAB	✓ 支持，等同于 应用设置
选项字节 TAB	✓ 支持，等同于写入选项字节
Program Memory TAB	✓ 支持，等同于编程 Program Memory 菜单
OTP Memory TAB (开发中)	✓ 支持
EEPROM TAB (开发中)	✓ 支持

表 3.1.3.6 当前 TAB 页写入支持的 TAB

注：“开发中”的功能表示仍然在开发当中，暂时还无法使用。

3.1.3.7“当前 TAB 页校验”按键

当前 TAB 页校验功能支持校验当前 TAB 页面的数据，详细的支持特性如图表 3.1.3.7 所示

TAB 名称	是否支持
烧录器设置 TAB	
选项字节 TAB	
Program Memory TAB	✓ 支持，等同于校验 Program Memory 菜单
OTP Memory TAB (开发中)	✓ 支持
EEPROM TAB (开发中)	✓ 支持

表 3.1.3.7 当前 TAB 页校验支持的 TAB

注：“开发中”的功能表示仍然在开发当中，暂时还无法使用。

3.1.3.8“自动编程”按键

自动编程按键等同于菜单项中的自动编程，功能完全一致，在此不做额外的补充说明

3.1.3.9“加载文件到当前 TAB 页”按键

加载文档到当前 TAB 页，是一个非常实用的功能，可以单独保存每一页 Tab 的配置数据和加载上次保存的数据，详细的支持特性如图表 3.1.3.9 所示

TAB 名称	是否支持
烧录器设置 TAB	✓ 支持从文件加载烧录器配置
选项字节 TAB	✓ 支持文件加载选项字节
Program Memory TAB	✓ 支持从文件加载整个 Program Memory
OTP Memory TAB (开发中)	✓ 支持
EEPROM TAB (开发中)	✓ 支持

表 3.1.3.9 从文件中加载 TAB 页

注意事项：

- “开发中”的功能表示仍然在开发当中，暂时还无法使用
- 烧录器设置从文件加载时，需要输入上次保存烧录器设置的文件密码，否则将无法加载(用户数据的安全性考虑)，格式为 pkg 格式
- 选项字节的加载支持 Power Writer 定义的 Hex,bin,S19 格式，非 Power Writer 导出的格式无法支持

- Program Memory 加载文件是加载到缓冲区，不显示分段信息

3.1.3.10“保存当前 TAB 页到文件”按键

保当当前 TAB 页到文档，是一个非常实用的功能，可以单独保存每一页 Tab 的配置数据和加载上次保存的数据，详细的支持特性如图表 3.1.3.10 所示

TAB 名称	是否支持
烧录器设置 TAB	✓ 支持保存到文件
选项字节 TAB	✓ 支持保存选项字节到文件
Program Memory TAB	✓ 支持保存整个 Program Memory 到文件
OTP Memory TAB (开发中)	✓ 支持
EEPROM TAB (开发中)	✓ 支持

表 3.1.3.9 从文件中加载 TAB 页

注意事项：

- “开发中”的功能表示仍然在开发当中，暂时还无法使用
- 烧录器设置保存到文件时，需要填写烧录器设置的文件密码(用户数据的安全性考虑)，格式为 pkg 格式
- 选项字节的保存支持 Power Writer 定义的 Hex,bin,S19 格式
- 保存 Program Memory 到文件是保存的是缓冲区，不显示分段信息，如果需要分段，请使用保存项目到文件，同样支持 Hex,bin,s19 格式。

3.1.4 Power Writer Tab 标签页

Power Writer 包含数个标签页，其中包含固定的三个标签页：烧录器设置和选项字节，以及 Program Memory 标签，以及包含两个动态的标签页（根据所选择的芯片自动识别）: OTP Memory 和 EEPROM 标签页，以下章节将详细介绍各个标签页的设置和使用方法。

3.1.4.1 烧录器设置 Tab 标签页

烧录器选项标签页作为 Power Writer 最重要的部分，包含了 Power Writer 数十项常用功能，本节内容非常重要，请用户仔细阅读其中的功能说明，才能让 Power Writer 发挥出应有的作用，烧录器选项 Tab 如图 3.1.4.1 所示

图 3.1.4.1 Power Writer 烧录器设置 Tab

烧录器设置 Tab 又分为：芯片设置、烧写功能设置、日志窗口三大部分，以下将对三大功能区域进行详细地介绍。

3.1.4.1.1 芯片设置

图 3.1.4.1.1 芯片设置界面

芯片设置部分包含芯片设置的常用基础功能，界面显示如图 3.1.4.1.1，包含:

- MCU 型号：显示用户当前选择的 MCU 型号，如：STM32F071xB，创芯工坊为了让 Power Writer 的功能性得到最大的发挥，针对所有芯片的细节，投入了大量的人力物力核对官方资料，分类汇总，而不是做成一个通用的 Flash Tools，Power Writer 更是一个全功能的开发工具。

  (注：芯片的分类格式以 Flash 容量为准，例如 STM32F071xB，x 表示可以代表任何的封装类型)

- 

选择芯片按钮：使用 Power Writer 前需要选择对应的芯片，Power Writer 会逐步增加常用芯片厂商芯片的适配，目前已适配完 STM32 全系列芯片，并对每个厂商芯片的更新进行追踪，确保适配厂家市面上可以买到的芯片在 Power Writer 中都能找到，点击选择芯片后将会弹出芯片选择对话框，左侧为芯片品牌分类，中间系列，右侧是芯片型号列表，非常直观地显示，如图 3.1.4.1.1 -2 所示

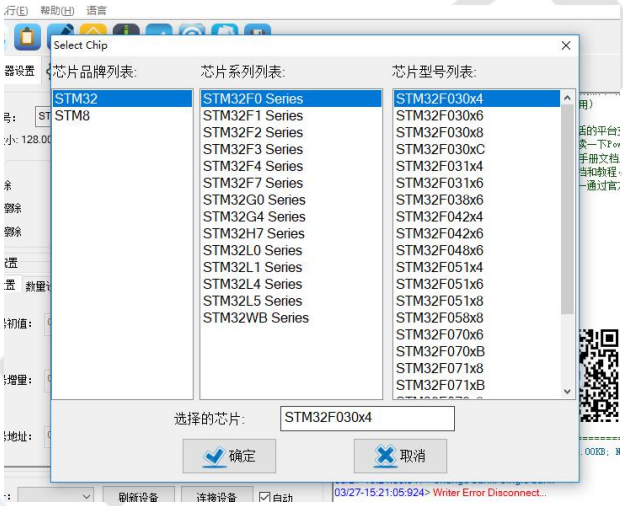


图 3.1.4.1.1 -2 选择芯片界面

- 

：当修改了 Power Writer 的相关设置，需要手动点击一次将配置同步到 Power Writer 硬件端。

  注意：
  - 当选择芯片时，会自动同步设置到烧录器
  - 当烧录器自动连接时，会自动同步设置到烧录器

应用设置之后，Power Writer 将重新连接目标芯片，并同步当前芯片的选项字节到 Power Writer 应用软件端，如下图 3.1.4.1.1 -3 所示

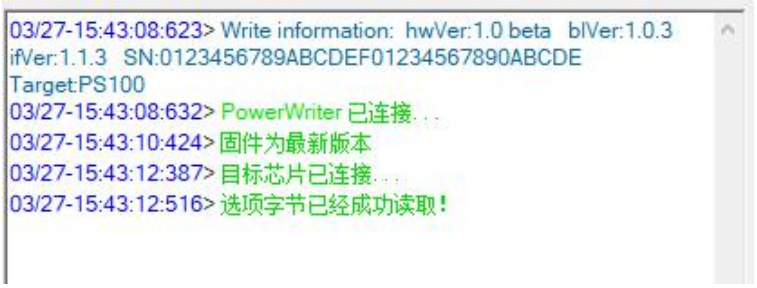


图 3.1.4.1.1 -3 应用芯片时将自动连接目标芯片并同步

- 选择完芯片后可以直观地看到目标芯片的 Flash、OTP、EEPROM 大小，如显示 Flash 大小：128.00KB NONE NONE 表示 Flash 为 128K，无 OTP 、EEPROM。(OTP、EEPROM 目前处于开发当中)。

- 擦除方式：擦除方式用于设置用于离线烧录时 Power Writer 的擦除方式，默认为全片擦除。

擦除方式	说明
不擦除	离线烧录时不对目标芯片进行擦除
Chip 擦除	离线烧录时对芯片进行全片擦除
页面擦除	离线烧录时对芯片进行 Sector 擦除(根据实际大小擦除)

表 3.1.4.1.1 -4 Power Writer 的离线擦除方式选择

注意：为何 Power Writer 没有 Bank 擦除？其实是 Power Writer 对 Bank 的已经做了自动化处理，无须用户选择 Bank 1 还是 Bank 2，无论是单 Bank 还是双 Bank，Memory Mapping 的地址为同一个地址，区别在于 Sector 表的大小不一样，而 Power Writer 支持完整的 Sector Class ,参考 Program Memory 的 Flash 扇区映射表即可看到 Bank 切换时，Sector 表的动态切换，Power Writer 对于 Bank 的擦除默认使用 Sector 擦除来替代。

- 接口电平：Power Writer 支持四种不同的电平匹配方式，见表 3.1.4.1.1 -5 ：

接口电平	说明
1.8 V	设置接口电平为 1.8V
3.3 V	设置接口电平为 3.3V
5.0 V	设置接口电平为 5.0V
外部输入	接口电平设置为根据外部参考电平

表 3.1.4.1.1 -5 Power Writer 接口电平设置为外部参考

- 其他：包含速度调节、选项字节更新方式、蜂鸣器选择方式：见下表 3.1.4.1.1-6 所示：

功能	说明
编程速度调节	10MHZ (默认)
	5MHZ
	2MHZ
	500KHZ
	200KHZ
	100KHZ
	50KHZ

	20KHZ
	10KHZ
	5KHZ
选项字节更新方式	烧录前无操作 -> 烧录后无操作
	烧录前无操作 -> 烧录后写入用户设置的 Option Byte
	烧录前恢复出厂默认值->烧录后无操作
	烧录前恢复出厂->烧录写入用户设置的 Option Byte
蜂鸣器	是否开启蜂鸣器提示音

表 3.1.4.1.1 -6 Power Writer 编程速度、选项字节更新方式、蜂鸣器选项

附加补充：为何速度没有 20MHZ，40MHZ 或者更高的速度？原因是：过高的编程速度，对生产环境有非常高的要求，在实际的测试中，20MHZ 的编程速度，会增加不良率，Power Writer 理论上任意速度调节，但是为了方便用户进行选择，我们进行了优化显示，提供常用的编程速度选择。

3.1.4.1.2 烧写功能配置

烧写功能配置用于配置 Power Writer 提供的众多附加功能，包括软件序列号的设置、离线烧录次数以及自动检测配置、信号输出控制、以及创芯工坊自主开发的多功能授权系统。

3.1.4.1.2.1 序列号设置

Power Writer 支持设置软件序列号，序列号长度为 4 字节，为了保持统一，序列号存储在 Flash 中，如需把 OPTION BYTE 中的 DATA0 DATA1 当做序列号，可以联系我们，序列号设置界面如图 3.1.4.1.2.1 - 1 所示

图 3.1.4.1.2.1 - 1 Power Writer 软件序列号设置

选项如下：

- 序列号初值：设置序列号的初始值，默认为 0x00000000
- 序列号增量：设置序列号的增量，默认为 0x00000001
- 序列号地址：设置序列号在 Flash 中的地址，默认为当前芯片的最末尾地址 - 4 的位置，如图 128Kb 的 STM32F071xB 芯片的，地址在 0x0801FFFC。
- 添加序列号功能：序列号功能的开关。
- 序列号 10 进制显示：切换序列号显示为 10 进制，默认为 16 进制
- 序列号大端模式：序列号使用大端模式写入，默认为小端模式

补充：大小端的资料请参考百度百科的描述，数据大小端百科。

3.1.4.1.2.2 数量设置和芯片检测

Power Writer 支持离线量产，设置离线量产的烧录次数，默认长度为 4 字节长度，界面显示如图 3.1.4.1.2.2 所示

图 3.1.4.1.2.2 Power Writer 数量设置和芯片检测

选项如下：

- 限制烧写次数：设置离线量产的次数，默认为 0x00000001，同时可以切换是否需要开启或者是关闭离线次数限制。
- 自动检测芯片：当开始此功能后，Power Writer 将自动检测芯片是否放入或者是芯片是否已拿走，无须再额外按烧录按键，或者是通过 CTRL 给信号。
- 芯片放入稳定时间：当 Power Writer 检测到芯片时，等待一段时间在执行烧录的后续操作，防止因为抖动而导致数据烧写失败或者是错误。
- 芯片拿开稳定时间：当 Power Writer 检测到芯片拿开时，等待一段时间再进行放入检测，防止因为抖动而导致对同一个芯片执行多次操作。
- 16 进制显示：切换显示模式，默认为 16 进制

补充：大小端的资料请参考百度百科的描述，数据大小端百科。

3.1.4.1.2.3 信号输出控制

Power 提供烧录时灵活的编程完之后信号输出控制，如下图 3.1.4.1.2.3 所示。

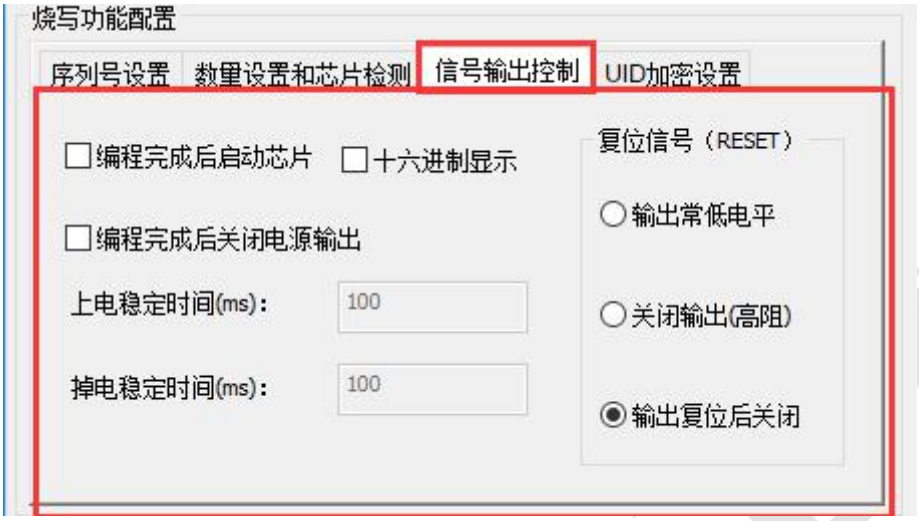


图 3.1.4.1.2.3 Power Writer 信号输出控制设置

选项包含：

- 编程完成后启动芯片：当编程完成之后，执行 Reset and Run
- 16 进制显示：16 进制显示上电稳定时间(ms)和掉电稳定时间(ms)
- 编程完成后关闭电源输出：当编程完成之后关闭电源输出，开启此选项后，每烧录完一颗芯片，都会断一下点，当下次启动烧录的时候，再根据上电稳定时间和掉电稳定时间执行供电。
- 上电稳定时间：当给芯片上电之后多少 ms 之后开始执行操作，默认为 100ms
- 掉电稳定时间：当给芯片断电之后至少多少 ms 之后开始执行下一次供电操作，默认为 100ms
- 复位信号：当烧录完芯片后，将会从 REST 引脚输出设定地输出信号类型，可以为如下三种
  - 输出常低电平：RESET 引脚一直输出低电平
  - 关闭输出：RESET 引脚切换为浮空输入模式,相当于禁用了此功能
  - 输出复位后关闭：此功能将产生一个 10ms 的低脉冲方波，然后拉高，让芯片产生一次复位动作.

注意：

1：(烧录完成后启动芯片、芯片自动检测) 和 烧录完关闭电源输出只能二选一，原因在于当关闭电源输出了，就无法启动芯片，也无法执行芯片自动检测；

2：烧录过程中的复位操作由 Power Writer 自动控制，大部分芯片都可以执行 under reset 复位功能，针对少部分芯片由于芯片本身的限制，需要硬件复位信号才可以复位，如果不清楚芯片是否可以执行软件复位，我们建议您在烧录芯片时将 RESET 引脚也接上。

3.1.4.1.2.4 UID 加密设置(核心功能)

UID 加密设置是创芯工坊研发的目前市面上烧录器在授权控制上最强大的加密版本，不但提供了随机的内置离线授权方法，一键生成动态的项目代码，同时提供了在线授权服务器（目前市面上提供此功能的只有 Segger 的 Secure Flash）支持第三方基于创芯工坊提供的授权服务器模板自建授权服务器，支持在线非对称授权方案，创芯工坊技术团队提供全程技术支持，支持离线 ICWKEY ECDSA 非对称授权算法，下面章节将详细介绍 UID 授权算法的核心功能,授权设置界面如图 3.1.4.1.2.4 所示。

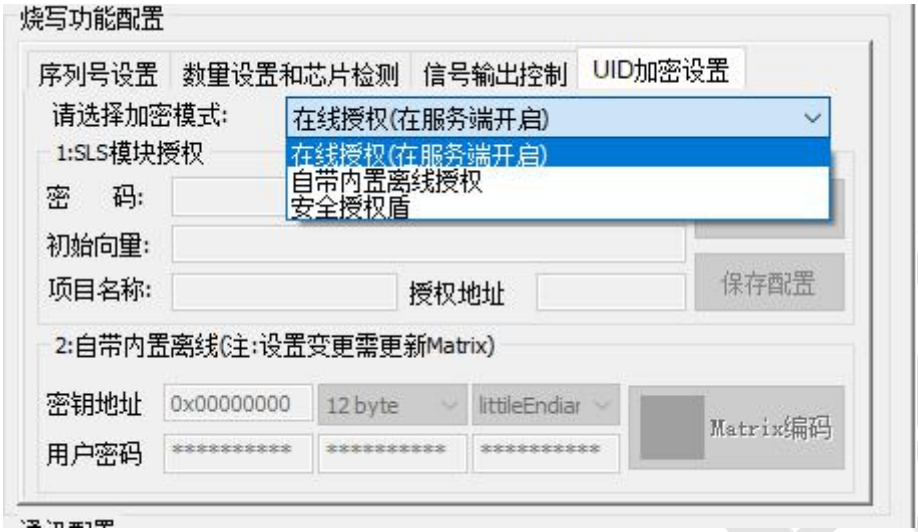


图 3.1.4.1.2.4 授权设置界面

3.1.4.1.2.4 服务器在线授权方案

在线授权方案功能由创芯工坊官方提供，此时的烧录器内部不存储离线固件，而是将固件提交到创芯工坊的后台管理控制台以订单形式发布，客户再通过创芯工坊客户端实现远程量产烧录，烧录芯片时需要全程联网，从授权服务器获取授权数据，在线授权方案同样是基于 CID 的，整个授权的算法由 Power Writer 用户/ 创芯工坊用户自主设计。具体的在线授权使用方法，请参考 License Server 章节/ 创芯工坊后台章节 / 以及在线授权方案的应用指导章节部分内容。本节做了一个概览介绍，为了更加充分的理解在线授权方案，下面演示了 Power Writer 的在线授权流程，如图 3.1.4.1.2.4 所示。

图 3.1.4.1.2.4 创芯工坊 Power Writer 在线授权流程

注：1：在线授权方案，Power Writer 端没有太多的设置选项，选为在线授权即可；
2：在线授权方案，需要依赖于创芯工坊的整套流程,自建服务器的话需要按照规范开发；
3：在线授权是一个可选项，如果 Power Writer 端选择在线授权,但是创芯工坊服务端没有设置在线授权，也不会有在线授权的功能,具体请参考创芯工坊服务器在线授权。

3.1.4.1.2.5 Power Writer 内置离线授权

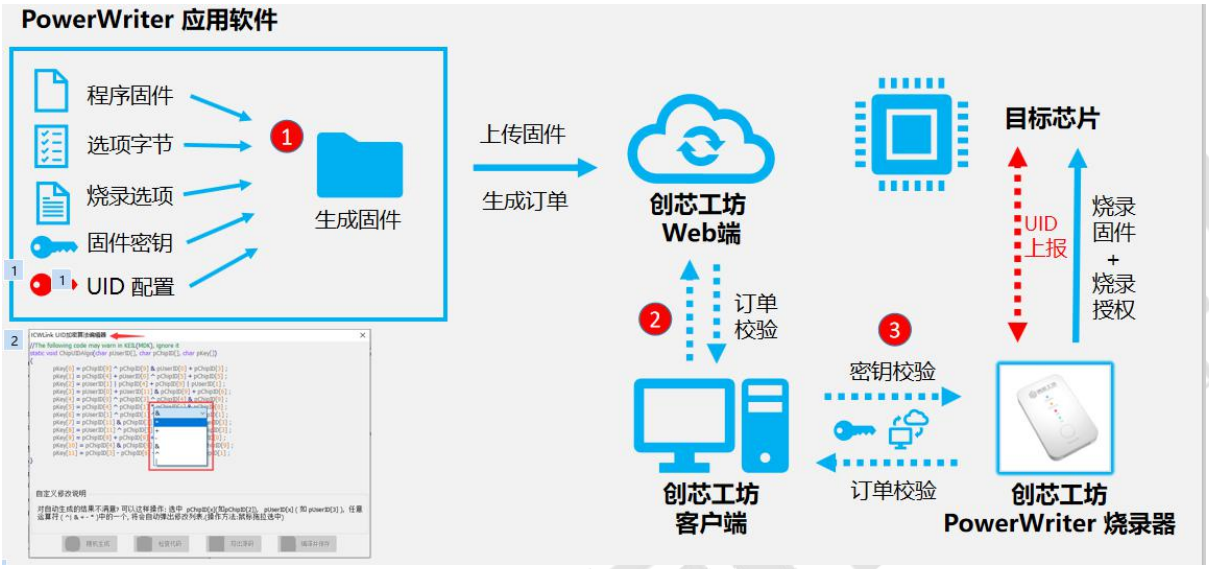


图 3.1.4.1.2.5 PowerWriter 内置离线授权流程(图中包含了创芯工坊)

Power Writer 内置了基于随机矩阵算法的 UID 离线授权方法，跟市面上固定授权方法不同的是 Power Writer 可以由开发者自由编辑算法矩阵。Power Writer 内置解析算法，对矩阵进行解析生成正确的算法，内置离线授权根据用户选择的芯片，参数设置自动生成 Demo 代码，极大地提高了了用户的开发效率，离线授权界面设置如图 3.1.4.1.2.5 -1 所示:

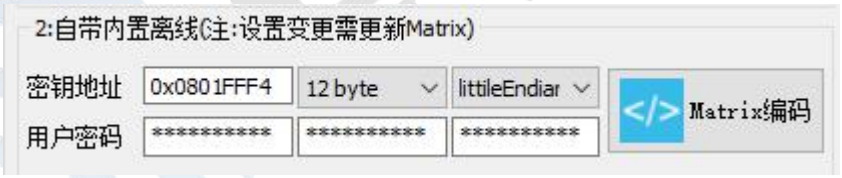


图 3.1.4.1.2.5 -1 Power Writer 内置离线授权设置界面

离线授权基础设置包含：

- 密钥地址：密钥地址可以理解为存放授权信息的地址，他的默认地址设定为 芯片 Flash 容量 - 12 的位置，上图是 STM32F071CB 的默认存储地址
- 用户密码长度：这里可以填写用户设定密码长度，默认为 12 字节，可选 4 字节，8 字节长度
- 数据存储模式：数据存储模式分为小端模式和大端模式
- 用户密码(3 组用户密码)：根据设定可以设定最多三种用户密码

  Matrix 编码：Matrix 编码定义了用户可以编辑的离线授权的加密矩阵，如下图 3.1.4.1.2.5 -2 所示。

图 3.1.4.1.2.5 -2 Power Writer 离线授权随机矩阵

Power Writer 提供了强大的随机矩阵授权算法，用户可以快速的随机生成功能，生成独一无二的随机授权矩阵验证算法，同时可以对随机算法矩阵的强度进行优化判断，自动导出 Demo 代码，

Power Writer 离线授权随机矩阵主要包含三个区域，如下所示：

- UID 加密算法编辑器代码编辑预览：此区域提供了代码预览和代码快速编辑的功能，通过随机生成代码或者手动修改算法矩阵，都可以实时显示在此区域，如需手动编辑算法矩阵，请参考自定义 UID 修改说明的操作流程。
- 自定义 UID 修改说明：自定义 UID 修改说明如下图所示，用户如果需要手动修改代码的组成结构，可以按照如下图所示操作。

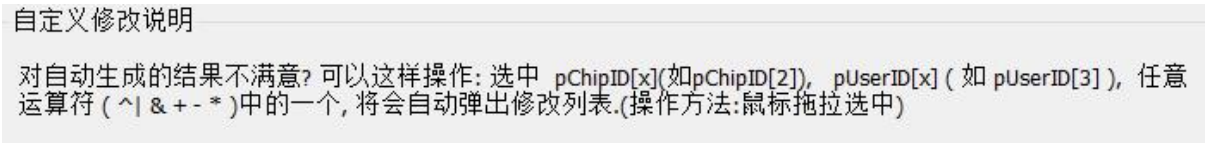


图 3.1.4.1.2.5 -3 Power Writer 离线矩阵自由编辑说明

图 3.1.4.1.2.5 -4 Power Writer 离线矩阵自由编辑演示

- 随机生成：随机生成功能将会生成随机矩阵数组，用户每次生成将会得到真正的授权矩阵算法。
- 检查代码：检查代码功能将会验证当前矩阵数组的强度，判断依据为是否每一个运算符都有参与运算、每一个 UID 和每一个用户 key 是否有有用到，检查提示截图如下图 3.1.4.1.2.5 -5 所示：

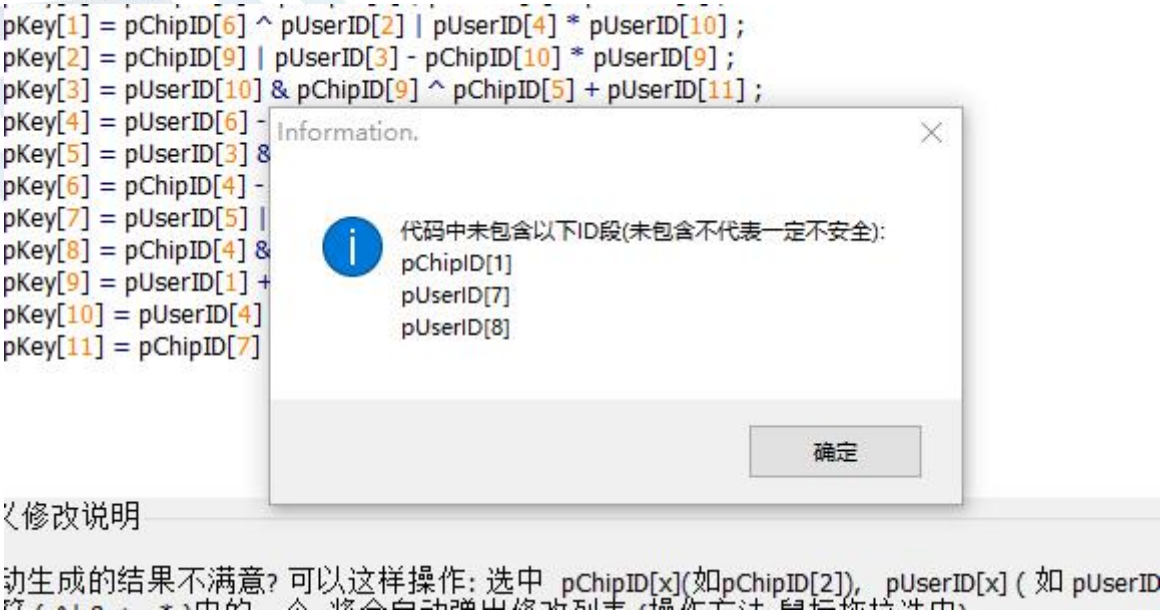


图 3.1.4.1.2.5 -4 代码检查

■ 导出源码：导出源码功能将用户设置好的信息，导出为 MDK Source Code(如果用户 IDE 为其他格式，请包含对应的源码和头文件即可使用)，导出的源码 Demo 如下图所示：

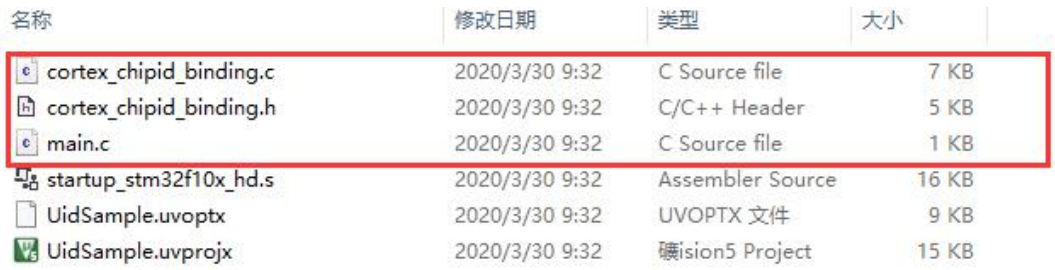


图 3.1.4.1.2.5 -4 UID 导出源码截图

生成的项目文件见下图所示:

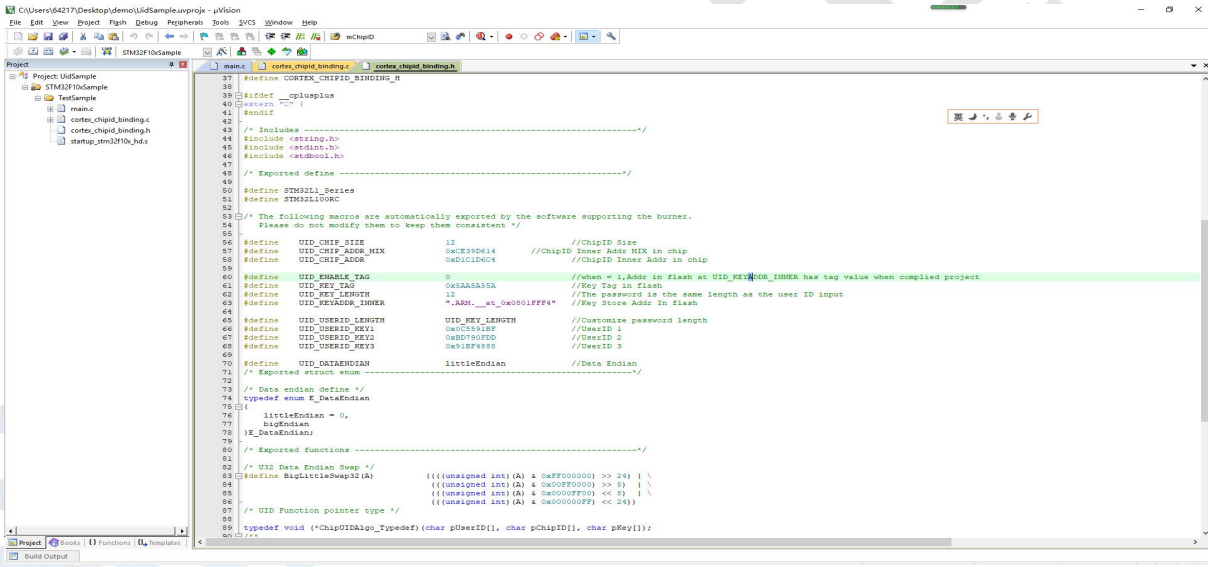


图 3.1.4.1.2.5 -4 UID 导出 Demo 截图

cortex_chipid_binding.c：芯片 UID 绑定代码源文件，以下是详细解释了 cortex_chipid_binding 中的代码组成结构,

1：Matrix 矩阵如下图格式，包含两种定义，一种加入了随机花指令，通过指令开启花指令，花指令的作用是在原始的矩阵中混入一些无关的代码，让反编译看到的结果和预想中有出入，增加分析难度，此方法对于芯片保护级别只有 0 和 1 两级的芯片尤其有用，用过可以通过以下开关进行开启或者关闭花指令

```
#define ENABLE_JUNK_CODE          1//Enable/Disable junk code
```

```
#if ENABLE_JUNK_CODE
static char mJunkCodeVar[UID_KEY_LENGTH];   //junkCode for mix

//The following code may warn in KEIL(MDK), ignore it
static void ChipUIDAlgo(char pUserID[], char pChipID[], char pKey[])
{
    pKey[0] = pUserID[5] & pUserID[10] - pChipID[4] | pChipID[10] ;
    pKey[1] = pChipID[8] | pUserID[8] | pUserID[1] * pChipID[2] ;
    mJunkCodeVar[9] = pUserID[10] - mJunkCodeVar[6] & mJunkCodeVar[7] & mJunkCodeVar[6] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[9] = pUserID[3] + mJunkCodeVar[10] ^ pUserID[4] ^ pKey[6] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[1] = mJunkCodeVar[5] | pKey[10] - pChipID[11] - pKey[8] ;//junk code,Your can remove or add your's junk code
    pKey[2] = pChipID[11] * pChipID[6] & pUserID[0] | pUserID[1] ;
    mJunkCodeVar[2] = pUserID[2] & pChipID[5] | pUserID[0] * pUserID[1] ;//junk code,Your can remove or add your's junk code
    pKey[3] = pChipID[4] - pUserID[9] - pUserID[10] ^ pChipID[4] ;
    mJunkCodeVar[7] = pUserID[3] + pChipID[7] * pUserID[1] & pUserID[4] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[7] = pKey[11] & pKey[7] & mJunkCodeVar[0] + pUserID[0] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[4] = mJunkCodeVar[5] ^ pUserID[3] | pKey[6] + pKey[10] ;//junk code,Your can remove or add your's junk code
    pKey[4] = pUserID[10] & pUserID[8] | pChipID[11] + pUserID[1] ;
    mJunkCodeVar[2] = pUserID[10] ^ pUserID[6] & pChipID[3] ^ pChipID[1] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[6] = mJunkCodeVar[6] * pKey[0] | pKey[1] * pKey[1] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[3] = pKey[7] - pUserID[4] - mJunkCodeVar[0] - pUserID[2] ;//junk code,Your can remove or add your's junk code
    pKey[5] = pUserID[5] * pUserID[11] | pChipID[0] + pUserID[5] ;
    mJunkCodeVar[2] = mJunkCodeVar[8] | pChipID[5] ^ mJunkCodeVar[11] * mJunkCodeVar[11] ;//junk code,Your can remove or add your's junk code
    pKey[6] = pUserID[6] + pUserID[5] - pUserID[1] | pUserID[6] ;
    mJunkCodeVar[8] = pChipID[7] & pUserID[0] - mJunkCodeVar[4] & pChipID[7] ;//junk code,Your can remove or add your's junk code
    pKey[7] = pChipID[6] + pUserID[1] | pUserID[1] ^ pUserID[0] ;
    mJunkCodeVar[0] = pUserID[10] | pKey[7] - pKey[1] & pChipID[8] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[4] = mJunkCodeVar[1] + pUserID[11] + pChipID[1] | mJunkCodeVar[1] ;//junk code,Your can remove or add your's junk code
    pKey[8] = pChipID[3] ^ pUserID[9] | pChipID[10] | pUserID[10] ;
    mJunkCodeVar[7] = pUserID[5] * pChipID[1] - pUserID[4] | pKey[11] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[8] = pChipID[8] & mJunkCodeVar[8] & pUserID[9] - pChipID[10] ;//junk code,Your can remove or add your's junk code
    pKey[9] = pChipID[4] & pUserID[4] ^ pUserID[5] - pUserID[5] ;
    mJunkCodeVar[1] = pChipID[0] * pUserID[11] * pChipID[8] + pKey[3] ;//junk code,Your can remove or add your's junk code
    mJunkCodeVar[5] = mJunkCodeVar[8] + pUserID[11] * mJunkCodeVar[0] ^ pUserID[7] ;//junk code,Your can remove or add your's junk code
    pKey[10] = pChipID[6] & pUserID[3] - pUserID[0] - pChipID[0] ;
    pKey[11] = pUserID[4] & pUserID[8] | pUserID[6] | pChipID[0] ;
}

#else
//The following code may warn in KEIL(MDK), ignore it
static void ChipUIDAlgo(char pUserID[], char pChipID[], char pKey[])
{
    pKey[0] = pUserID[5] & pUserID[10] - pChipID[4] | pChipID[10] ;
    pKey[1] = pChipID[8] | pUserID[8] | pUserID[1] * pChipID[2] ;
    pKey[2] = pChipID[11] * pChipID[6] & pUserID[0] | pUserID[1] ;
    pKey[3] = pChipID[4] - pUserID[9] - pUserID[10] ^ pChipID[4] ;
    pKey[4] = pUserID[10] & pUserID[8] | pChipID[11] + pUserID[1] ;
    pKey[5] = pUserID[5] * pUserID[11] | pChipID[0] + pUserID[5] ;
    pKey[6] = pUserID[6] + pUserID[5] - pUserID[1] | pUserID[6] ;
    pKey[7] = pChipID[6] + pUserID[1] | pUserID[1] ^ pUserID[0] ;
    pKey[8] = pChipID[3] ^ pUserID[9] | pChipID[10] | pUserID[10] ;
    pKey[9] = pChipID[4] & pUserID[4] ^ pUserID[5] - pUserID[5] ;
    pKey[10] = pChipID[6] & pUserID[3] - pUserID[0] - pChipID[0] ;
    pKey[11] = pUserID[4] & pUserID[8] | pUserID[6] | pChipID[0] ;
}

#endif
```

图 3.1.4.1.2.5 -4 UID 导出源码 Matrix 花指令和原始指令截图

U32DataEndiaSwap：32 位数据大小端交换函数

```
/**
  * @brief  Data Endian Swap in uint32
  * @note   Please keep the burner and project consistent, do not modify it
  * @retval None
  */

void U32DataEndiaSwap(uint32_t * pBuffer, uint32_t size)
{
    int i;
    for(i = 0; i < size; i ++){
        pBuffer[i] = BigLittleSwap32(pBuffer[i]);
    }
}
```

图 3.1.4.1.2.5 -4 UID 导出源码 32 位数据大小端交换函数

```
/* store full chipID */
static char mChipID[UID_KEY_LENGTH];

/* initial chip id */
static uint32_t mChipAddr = UID_CHIP_ADDR ^ UID_CHIP_ADDR_MIX;                //global

/* calc key compare with  mKey */
static char mCalcKey[UID_KEY_LENGTH];


/* store full userID */
static char mUserID[] = {
#if (UID_USERID_LENGTH == 4)
    (UID_USERID_KEY1 & 0xff),((UID_USERID_KEY1>>8) & 0xff),((UID_USERID_KEY1>>16) & 0xff),((UID_USERID_KEY1>>24) & 0xff),
#elif (UID_USERID_LENGTH == 8)
    (UID_USERID_KEY1 & 0xff),((UID_USERID_KEY1>>8) & 0xff),((UID_USERID_KEY1>>16) & 0xff),((UID_USERID_KEY1>>24) & 0xff),
    (UID_USERID_KEY2 & 0xff),((UID_USERID_KEY2>>8) & 0xff),((UID_USERID_KEY2>>16) & 0xff),((UID_USERID_KEY2>>24) & 0xff)
#else
    (UID_USERID_KEY1 & 0xff),((UID_USERID_KEY1>>8) & 0xff),((UID_USERID_KEY1>>16) & 0xff),((UID_USERID_KEY1>>24) & 0xff),
    (UID_USERID_KEY2 & 0xff),((UID_USERID_KEY2>>8) & 0xff),((UID_USERID_KEY2>>16) & 0xff),((UID_USERID_KEY2>>24) & 0xff),
    (UID_USERID_KEY3 & 0xff),((UID_USERID_KEY3>>8) & 0xff),((UID_USERID_KEY3>>16) & 0xff),((UID_USERID_KEY3>>24) & 0xff)
#endif
};

/* Store the key calculated by the burner   */
const static char mKey[UID_KEY_LENGTH] __attribute__((section(UID_KEYADDR_INNER))) =
{
#if ENABLE_TAG
    #if (UID_KEY_LENGTH == 4)
        UID_KEY_TAG & 0xff,(UID_KEY_TAG >> 8) & 0xff,(UID_KEY_TAG >> 16) & 0xff,(UID_KEY_TAG >> 24) & 0xff
    #elif (UID_KEY_LENGTH == 8)
            UID_KEY_TAG & 0xff,(UID_KEY_TAG >> 8) & 0xff,(UID_KEY_TAG >> 16) & 0xff,(UID_KEY_TAG >> 24) & 0xff,
            UID_KEY_TAG & 0xff,(UID_KEY_TAG >> 8) & 0xff,(UID_KEY_TAG >> 16) & 0xff,(UID_KEY_TAG >> 24) & 0xff
    #else
            UID_KEY_TAG & 0xff,(UID_KEY_TAG >> 8) & 0xff,(UID_KEY_TAG >> 16) & 0xff,(UID_KEY_TAG >> 24) & 0xff,
            UID_KEY_TAG & 0xff,(UID_KEY_TAG >> 8) & 0xff,(UID_KEY_TAG >> 16) & 0xff,(UID_KEY_TAG >> 24) & 0xff,
            UID_KEY_TAG & 0xff,(UID_KEY_TAG >> 8) & 0xff,(UID_KEY_TAG >> 16) & 0xff,(UID_KEY_TAG >> 24) & 0xff
    #endif
#endif
};
```

图 3.1.4.1.2.5 -4 UID 导出源码存储变量定义

mChipID： 用于存储芯片的 UID，在启动的时候需要初始化一次，针对部分芯片 UID 不连续的，在初始化的时候合并成连续的数据存储

mChipAddr：用于存储混淆过的芯片 UID 地址信息，混淆过后无法在二进制代码中找到芯片的 ID 地址信息。

mCalcKey：用于存储动态 Matrix 编码的计算的结果，也就是存储正确密码

mUserID：存储用户设定的密码信息

mKey：存储 Power Writer 量产时写入的正确密码数据，在代码动态计算如果 mCalcKey 和 mKey 不匹配，说明此代码被非法拷贝,从而实现拒绝运行的目的。

void ChipUIDInitial()：此函数用于初始化 UID 信息，为了隐藏芯片的真实 UID ，PowerWriter 在导出源代码时，会将 UID_CHIP_ADDR 与 随机数 UID_CHIP_ADDR_MIX 进行异或运算，使其在编译好的 Bin 文件 hex 中不会显示真实的 ID 地址，然后在 ChipUIDInitial 中在动态初始化，将正确的 UID 存储在内存中，此函数分为以下几个功能：

1：在代码运行时 计算 UID 地址

2：将 UID 地址安全转移到另外一个变量。

3：检查 UID 转移代码是否被非法篡改或者无效地址

4：对于连续 UID 地址和 非连续 UID 地址拷贝到同一个 UID 数组中

ChipUIDInitial 的代码截图如图 3.1.4.1.2.5 -5 所示

```
160   */
161  void ChipUIDInitial()
162  {
163    //restore real id Addr
164
165    mChipAddr = UID_CHIP_ADDR;
166    mChipAddr ^= UID_CHIP_ADDR_MIX;
167
168    //Do not read directly from the address,you could use your's style
169    uint32_t chipIDAddr =0;
170    uint32_t mask = 0x0000000f;
171    for(int i = 0;i<sizeof(uint32_t) * 2;i++){
172      chipIDAddr |= mChipAddr  & (mask << i *4);
173    }
174    //if it has been modified by decompiling, exit
175    if(chipIDAddr == 0 || chipIDAddr == 0xffffffff)
176      return;
177    //copy to array
178  #ifdef STM32L1_Series   //offset 0x00,0x04,0x14
179    for(int i=0;i<UID_CHIP_SIZE;i++){
180      mChipID[i] = *(uint8_t *)chipIDAddr;
181      if(i != 7){
182        chipIDAddr++;
183      }
184      else{
185        chipIDAddr += 0x0D;
186      }
187    }
188  #else
189    for(int i=0;i<UID_CHIP_SIZE;i++){
190      mChipID[i] = *(uint8_t *)chipIDAddr;
191      chipIDAddr++;
192    }
193  #endif
194  }
195
```

图 3.1.4.1.2.5 -5 UID 初始化函数

ChipUIDAlgo_Calc: 用于计算 UID Matrix 矩阵计算，这里并非直接调用 Matrix 矩阵算法，而是通过函数指针间接调用函数，目的是，如果逆向分析人员通过分析得到了矩阵函数的地址，但是由于没有直接调用的代码，逆向分析无法确定 调用 ChipUIDAlgo 的父级函数，起到隐藏代码的目的。

```
91
92   char * ChipUIDAlgo_Calc()
93   {
94     ChipUIDAlgo_Typedef func = ChipUIDAlgo;
95     (*func)(mUserID,mChipID,mCalcKey);
96     if(UID_DATAENDIAN == bigEndian){
97       U32DataEndiaSwap((uint32_t *)mCalcKey,sizeof(uint32_t));
98     }
99     return mCalcKey;
100  }
101
102 /**
```

图 3.1.4.1.2.5 -5 UID 矩阵计算函数

ChipUIDAlgo_Check：此函数用于验证动态计算的结果和 Power Writer 的结果是否完全一致。

```
  bool __inline ChipUIDAlgo_Check()
  {
    ChipUIDAlgo_Calc();
    return  0 == memcmp(mCalcKey,mKey,UID_KEY_LENGTH);
  }
```

图 3.1.4.1.2.5 -6 UID 验证函数

cortex_chipid_binding.h：芯片 UID 绑定代码头文件，其定义信息如图 3.1.4.1.2.5 -7 标注所示。

```
#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include <string.h>
#include <stdint.h>
#include <stdbool.h>

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

#define STM32F0_Series
#define STM32F071xB

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

#define    UID_CHIP_SIZE               12                      //ChipID Size
#define    UID_CHIP_ADDR_MIX           0xE4E23C7A              //ChipID Inner Addr MIX in chip
#define    UID_CHIP_ADDR               0xFB1DCBD6              //ChipID Inner Addr in chip

#define    UID_ENABLE_TAG              0                       //when = 1,Addr in flash at UID_KEYADDR_INNER has tag value when complied project
#define    UID_KEY_TAG                 0x5AA5A55A              //Key Tag in flash
#define    UID_KEY_LENGTH              12                      //The password is the same length as the user ID input
#define    UID_KEYADDR_INNER           ".ARM.__at_0x0801FFF4"  //Key Store Addr In flash

#define    UID_KEYADDR_PLACEHOLDER_EN  0                       //Key Store Addr In Flash Enable/Disable Placeholder
#define    UID_KEYADDR_MIX             0xF4A77775              //Key Store Addr In Flash Mix Rand
                                                               //Key Store Addr In Flash Mixed
#define    UID_KEYADDR_INNER_MIXED     (0x0801FFF4 ^ UID_KEYADDR_MIX)

#define    UID_USERID_LENGTH           UID_KEY_LENGTH          //Customize password length
#define    UID_USERID_KEY1             0x097FA3EF              //UserID 1
#define    UID_USERID_KEY2             0x438A3070              //UserID 2
#define    UID_USERID_KEY3             0xC3F2AB5E              //UserID 3

#define    UID_DATAENDIAN              littleEndian            //Data Endian
/* Exported struct enum ------------------------------------------------------*/
```

图 3.1.4.1.2.5 -7　UID 头文件定义

```
#define STM32F0_Series                  //当前芯片的系列
#define STM32F071xB                     //当前芯片的型号
#define UID_CHIP_SIZE                12 //当前芯片 ID 的长度，默认都是 12 字节
#define UID_CHIP_ADDR_MIX        0xE4E23C7A //用于隐藏芯片 ID 地址在代码中的位置
#define UID_CHIP_ADDR            0xFB1DCBD6     //芯片 ID 在内存中的地址
#define UID_ENABLE_TAG            0   //当为 1 时,将会对有占位符的授权地址填充 UID_KEY_TAG
#define UID_KEY_TAG              0x5AA5A55A     //有占位符标记
#define UID_KEY_LENGTH        12                  //用户输入的密码长度
#define UID_KEYADDR_INNER     ".ARM.__at_0x0801FFF4"     //授权数据再内存中的地址
#define UID_KEYADDR_PLACEHOLDER_EN     0          //是否在代码中开启授权数据占位符
#define UID_KEYADDR_MIX                0xF4A77775          //授权地址混淆随机数
#define UID_KEYADDR_INNER_MIXED    (0x0801FFF4 ^ UID_KEYADDR_MIX)     //授权数据存储地址
#define UID_USERID_LENGTH              UID_KEY_LENGTH          //用户密码长度
#define UID_USERID_KEY1                0x097FA3EF              //用户密码 1
#define UID_USERID_KEY2                0x438A3070              //用户密码 2
#define UID_USERID_KEY3                0xC3F2AB5E              //用户密码 3
#define       UID_DATAENDIAN               littleEndian            //数端序定义
```

main.c：测试 chipid 授权，此部分包含一个初始化和验证结果,用户再实际代码中的位置可以藏于任意代码位置。

```
void SystemInit()
{
  //don't call ChipUIDInitial() at there
  //...
}

int main()
{
  //Initial Chip
  ChipUIDInitial();

  //user code
  //......

  //Check in your code
  bool ret = ChipUIDAlgo_Check();
  if(ret){
    //ok
  }else{
    //false
  }
}
```

图 3.1.4.1.2.5 -8　Power Writer 离线授权验证流程演示

startup_stm32f10x_hd.s：芯片启动初始化代码，这里是 f10x 的测试

Uid*.uv*：MDK 项目文件

■ 编译并保存：编译并保存功能将用户设置好的 Power Writer 内置授权功能同步到 PowerWriter 硬件,同时导出源码供用户使用。

特别注意：

1. 如果修改了 密钥地址、长度、端序、或者是用户密码信息、需要重新打开 Matrix 编辑器，重新点击编译并保存，否则会出现信息不同步；
2. 生成的 Demo 中实际需要使用的只有 cortex_chipid_binding.c 和 cortex_chipid_binding.h，其他文件为辅助文件；
3. 如果 Power Writer 的用户有丰富的逆向分析经验，可以将生成的代码继续优化，加入花指令，或者是优化流程。

3.1.4.1.2.6 ICWKEY 授权工具进行授权

Power Writer 除了远程服务器在线授权、内置随机 Matrix 随机授权算法授权之外，此外提供了专业的授权工具 ICWKEY，通过 ICWKEY 可以实现非对称 ECDSA (ECC 非对称加密算法)授权，可以灵活将授权功能和 Power Writer 进行分离控制，并提供高强度的授权方法终极解决方案，针对高端产品，大批量产品授权，实现灵活的授权控制方案，ICWKEY 授权方法整个授权流程如图 3.1.4.1.2.6-1 所示。

Power Writer & ICWKEY 授权方案需要用户基于 ICWKEY 的 SDK 和开发文档进行二次开发，ICWKEY 单独提供灵活的授权控制，以及完整的 SDK 、文档和 Sample Code。具体参考 《ICWKEY 用户参考手册 RM0002.pdf》。

本节将概要介绍 ICWKEY 在 Power Writer 端配置，ICWKEY 配置工具和 ICWKEY SDK 二次开发的基础流程。

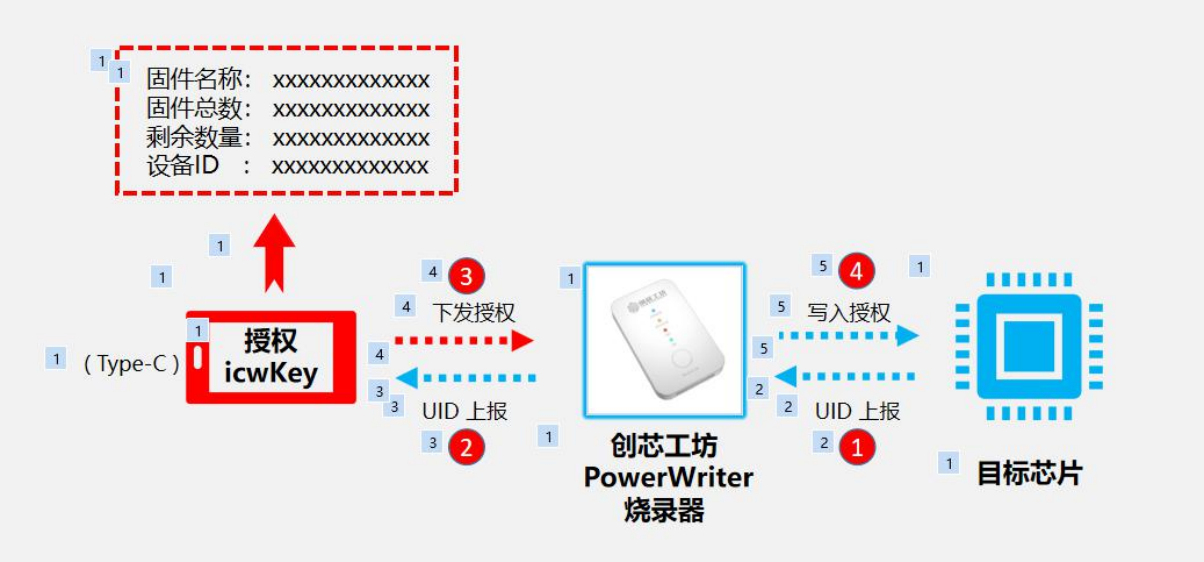


图 3.1.4.1.2.6-1 Power Writer +ICWKEY 授权工具进行芯片量产授权

ICWKEY 在 Power Writer 端配置：Power Writer 的 ICWKEY 配置功能如图 3.1.4.1.2.6-2 所示：

图 3.1.4.1.2.6-2 Power Writer 端 ICWKEY 通信配置

- 密码： 为了提供最高强度的通讯加密，Power Writer 与 ICWKEY 的通讯采用 AES128 CBC 模式加密，密码配置在 Power Writer 端随机生成，当密码框无焦点时默认不显示密码，密码可以使用随机生成功能进行生成.不提供手动填入。
- 初始向量： Power Writer 和 ICWKEY 的通讯除了 通信密码，同时提供一组初始向量，再通过创芯工坊的滚码算法，实现高强度的加密。
- 项目名称： 此名称将和 ICWKEY 的屏幕显示项目名称保持一致，默认格式为 :SafeLic_XXXXXXXX，如用户对默认的显示项目名称不满意。可以手动填写，默认最多为 16 个字节。
- 授权地址： 填写 ICWKEY 在 Flash 中的授权地址，Power Writer 将根据用户填写的此地址，写入授权信息到 目标芯片的 Flash 地址中。开发方法请参考 《ICWKEY 用户参考手册 RM0002.PDF》，此地址的默认值为芯片 Flash 的末尾 - 0x80 的位置。在基于 ICWKEY 开发完成项目后基于 MDK 导出的 Mapping 信息找到授权的地址，在此处填写当前正确的授权地址信息。
- 随机生成： 点击此按钮 Power Writer 将随机生成 **密码、初始向量、项目代码**。
- 保存配置： 当用户完成设置后，点击保存，此时 ICWKEY 配置信息将会保存到缓冲区。(注: 此时点保存并非同步到 Power Writer，同步到烧录器的方法统一通过离线加载并保存按钮菜单进行操作)。

ICWKEY 端配置：ICWKEY 提供了灵活的权限控制与授权控制，应用软件界面如下图所示。

图 3.1.4.1.2.6-3 ICWKEY 应用配置软件

- 项目名称：同 Power Writer 端的项目名称,请保持一致
- 密码：同 Power Writer 端的密码，请保持一致， 默认密码为 303132333435363738394142434445 46 等同于字符串的“0123456789ABCDEF”。
- 向量：同 Power Writer 端的向量，请保持一致，
- 选择设备：连上 ICWKEY 之后刷新设备即可看到设备
- 刷新设备：连上 ICWKEY 之后刷新下设备
- 连接设备：连接 ICWKEY

设备配置：

- 新项目名称：ICWKEY 默认的项目名称为 ICWorkShop，需要改成和 Power Writer 端一致
- 新密码：初次使用时，需要给 ICWKEY 配置新的通讯密码，否则进行授权时将会收到错误提示。
- 新向量：初次使用时，需要给 ICWKEY 配置新的通讯向量，否则授权是将会收到错误提示。
- 设备序列号：设定 ICWKEY 的序列号
- UID 最小值：开启限制 UID 烧录范围时有效，限定 UID 的授权范围
- UID 最大值：开启限制 UID 收录范围时有效，限定 UID 的授权范围
- 授权数量：可以授权的芯片数量
- 可配置次数：默认 ICWKEY 可以供 10 个项目使用，在首次配置时设置，后面不能再更改。
- 实际剩余次数：ICWKEY 还剩多少次可以用于配置
- 使能授权工具：可以开启或者关闭授权功能

- 允许固件升级：可以开启或者关闭固件升级功能
- 限制 UID 授权范围：开启或者关闭 UID 范围限制
- 允许写入 UID 算法：开启或者关闭 UID 算法写入
- 开启日志记录：开启或者关闭 ICWKEY 的日志记录功能

UID 算法：

- 向量矩阵加密：功能同 Power Writer 端内置的 Matrix 授权功能
- ECDSA 签名算法：ICWKEY 和 Power Writer 内置了,非对称加密算法 ECC，采用高性能的 mbedtls 技术框架，进行高度优化。产生签名的时间约为 300ms，验签的时间大约为 700ms。
- 用户自定义：此模式下需要获得创芯工坊的授权，针对大客户，创芯工坊将开放 ICWKEY 配置软件、协议文档、ICWKEY 硬件端源码、以及相关 SDK 资料。

日志信息：

ICWKEY 提供读取授权日志的功能，用于读取已授权的次数信息

测试授权：

ICWKEY 提供测试授权的功能，用于测试授权的性能

日志栏：

通过日志栏，可以实时看到 ICWKEY 的日志信息，和 Power Writer 一样，提供分色显示的日志浏览，方便用户进行查阅。

补充：

ICWKEY 作为专业的授权工具，由于采用创芯工坊内建格式的 ECDSA 签名算法，通信理论上可以无须加密也是安全的。但是为了保护用户的所有信息。我们依然对通信数据进行了混合的高强度加密算法。

ICWKEY 进行专业级别的授权时，请确保 Power Writer 端和 ICWKEY 端配置信息一致。

ICWKEY 的详细开发教程需额外参考 ICWKEY 的用户参考手册。

ICWKEY 自定义授权模式需要获得创芯工坊的授权，如有需求，请联系创芯工坊。

3.1.4.1.2.6 芯片 UID 的不完全列表

Power Writer 在数据中已经将所有的芯片的 UID 存储，在用户使用时将自动导出芯片的 UID 地址信息和项目代码，**用户无须过多的关心芯片 UID 地址**，如果用户需要查看芯片的 UID 信息，请参考附录 芯片 UID 地址信息汇总

3.1.4.1.3 通信配置

Power Writer 的通讯配置界面如下图 3.1.4.1.3 -1 所示：

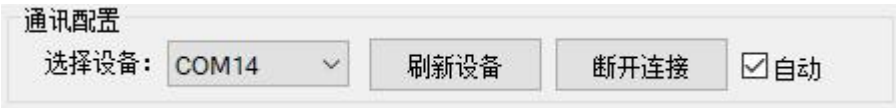


图 3.1.4.1.3 -1 PowerWriter 通信配置

- 选择设备列表：此列表显示当前扫描到的 Power Writer 硬件设备
- 刷新设备：用户可以手动点击此按钮来刷新设备列表。

- 断开连接：用户可以手动点击断开连接，断开与 Power Writer 的通讯
- 自动：默认勾选，将自动管理 Power Writer 设备的连接和断开，当插入 Power Writer 时，尝试自动连接到 Power Writer，当拔掉 Power Writer 时，将自动断开连接

Power Writer 首次连接成功时，将会自动执行固件检查，获取并显示 Power Writer 信息，如图 3.1.4.1.3 -2 所示

图 3.1.4.1.3 -2 Power Writer 自动执行固件检查并显示 Power Writer 信息

3.1.4.1.4 日志窗口

Power Writer 提供详细的日志操作列表，并使用不同颜色显示，日志窗口界面如图 3.1.4.1.4-1 所示

图 3.1.4.1.4-1 Power Writer 日志窗口

- 黑色：代表固定信息，如显示创芯工坊的官方联系方式
- 浅蓝色：代表一般性信息，如显示 Power Writer 的硬件信息

- 绿色：代表操作成功信息，比如烧录成功
- 浅红色：代表警告信息，或者是需要注意的信息
- 红色：代表错误信息，看到红色信息时，需要特别流程输出的内容，只有当遇到比较严重的错误才会出现此信息。

Power Writer 的日志功能额外提供一个右键菜单，见图 3.1.4.1.4-2 所示，提供清除日志，和拷贝日志功能。

- 清除日志：点击此菜单将清除 Power Writer 的所有操作日志
- 复制日志到剪贴板：将日志以富文本格式拷贝到剪贴板，用户可以粘贴到记事本,或者是粘贴到 Doc 文档，当粘贴到 Doc 文档时，将保留日志的原始排版信息。见图 3.1.4.1.4-3 所示。

3.1.4.1.4-2 Power Writer 日志窗口菜单

Power Writer Doc 文档日志：

3.1.4.1.4-3 Power Writer 提供 Doc 日志功能

3.1.4.2 选项字节 Tab 标签页

Power Writer 提供完整的选项字节定义，所有数据均来自官方提供的 Data Sheet，选项字节窗口的显示如图 3.1.4.2-1 所示。

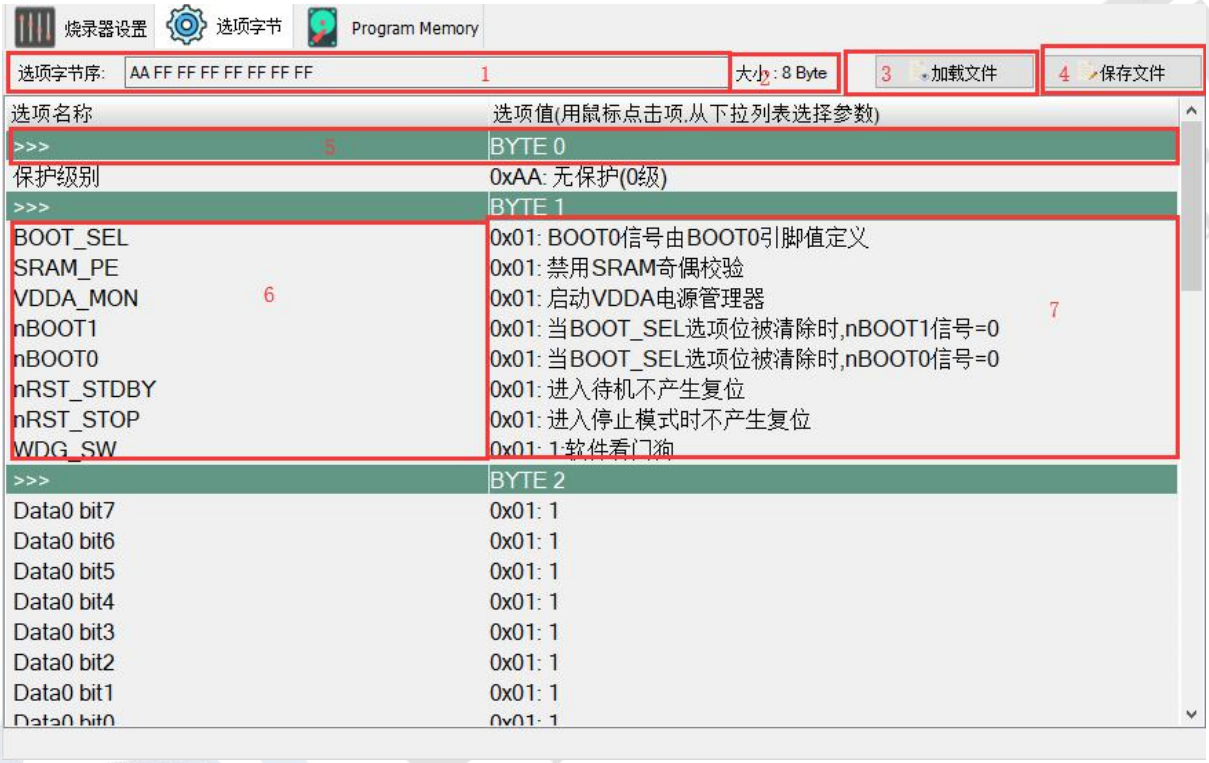


图 3.1.4.2-1 选项字节界面

Power Writer 提供每一个芯片的选项字节映射表，不仅提供选项字节的定义，并提供芯片的默认值设定，极大地方便了开发者，选项字节支持在线读写，并且支持有锁芯片的选项字节读写。从图 3.1.4.2-1 可以看出，Power Writer 的选项字节主要分为：

- **选项字节预览区：**选项字节预览区实时显示当前的选项字节设定，极大的方便开发者进行操作，为了确保用户的设定不会输入非法数据，Power Writer 暂不提直接修改选项字节的功能。
- **选项长度显示：**选项长度显示了当前选项芯片选项字节的字节长度，如 大小：8 Byte 所示。
- **选项字节保存到文件：**选项可以保存到文件，支持 hex，s19，bin 格式。
- **选项字节从文件加载：**选项字节也可以从已保存的文件中加载，支持 hex，s19，bin 格式。
- **选项字节 Byte 指示器：**Byte 只是器如 >>> BYTE 0 所示，Byte 指示器直观地将选项字节序按照指示器分开，并用底色显示。
- **选项字节设置项：**选项字节设置项位于选项字节窗口的左侧，显示名称和 Data Sheet 上保持一致。
- **选项字节设置值：**选项字节设置的值，通过鼠标点击设置项的值，将会自动弹出选项字节设置列表，列表中显示当前选项支持设置的值，如图 3.1.4.2-2 所示

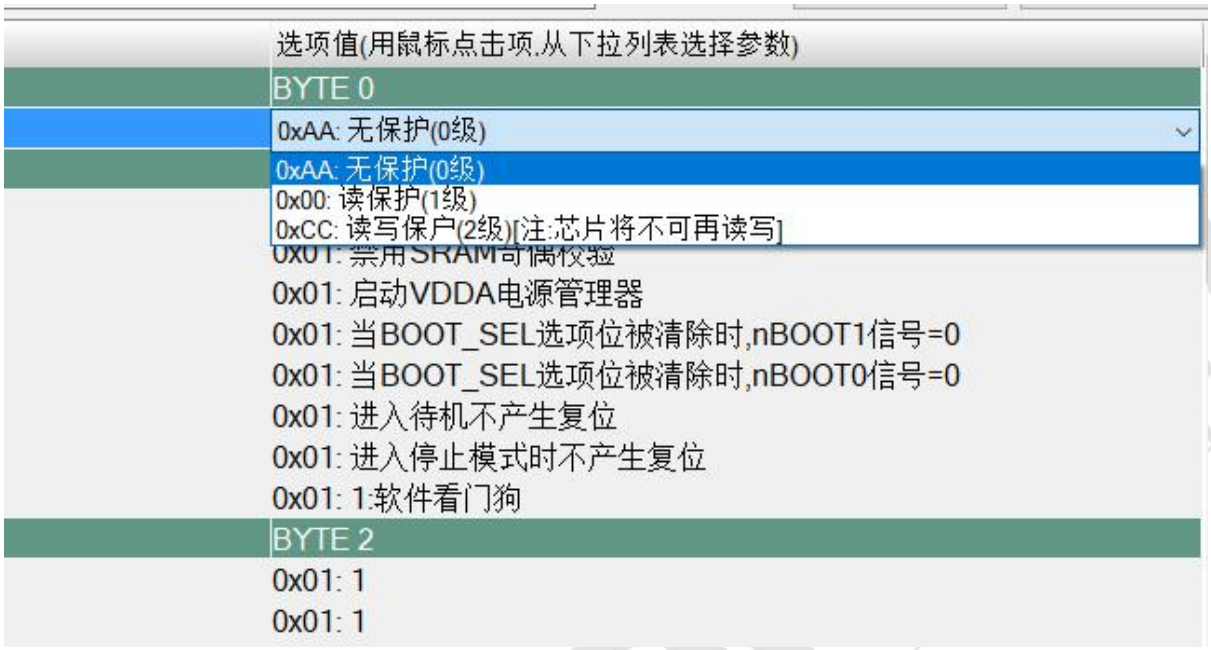


图 3.1.4.2-2 选项字节更改方法

附加补充：

- **选项字节的读取：**通过菜单->执行->读取选项字节，或者在选项字节窗口点击读取当前页工具栏按钮执行读取操作。
- **选项字节的写入：**通过菜单->执行->写入选项字节，或者在选项字节窗口点击写入当前页工具栏按钮执行写入操作。
- **选项字节的保存：**选项字节窗口点击**保存文件**，或者在选项字节窗口点击保存当前页工具栏按钮执行选项字节的保存操作。
- **选项字节的加载：**选项字节窗口点击**加载文件**，或者在选项字节窗口点击加载当前页工具栏按钮执行选项字节的加载操作。
- **选项字节的修改：**用鼠标点击设置项的选项值从下来列表进行操作。
- **选项字节的双 Bank：**Power Writer 支持自动识别双 Bank，当前芯片为 STM32G483xE，如图 3.1.4.2-3 所示，可以看到双 Bank 模式下的 Sector 分区表为每一个扇区为 2KB，当切换为 4K，再核对芯片手册 RM0440.PDF,可以看出和芯片手册保持一致。

选项名称	选项值(用鼠标点击项,从下拉列表选择参数)
>>>	BYTE 0
保护级别	0xAA: 无保护(0级)
>>>	BYTE 1
nRST_SHDW	0x01: 进入关机模式时不产生复位
nRST_STDBY	0x01: 进入待机不产生复位
nRST_STOP	0x01: 进入停止模式时不产生复位
BOR_LEV	0x00: BOR 0级. 复位电平阈值 1.7 V
>>>	BYTE 2
nBOOT1	0x01: nBOOT1设置为高
DBANK	0x01: 双bank模式与64位数据
BFB2	0x00: 双Bank引导禁用
WWDG_SW	0x01: 软件看门狗
IWDG_STBY	0x01: IWDG计数器在待机状态下工作

图 3.1.4.2-3 Power Writer 的双 bank 切换

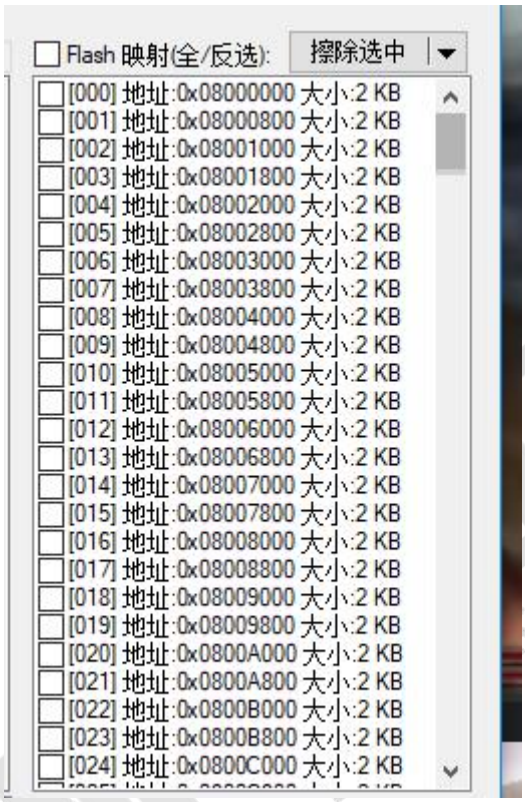


图 3.1.4.2-4 双 bank 下的 Sector 分区表

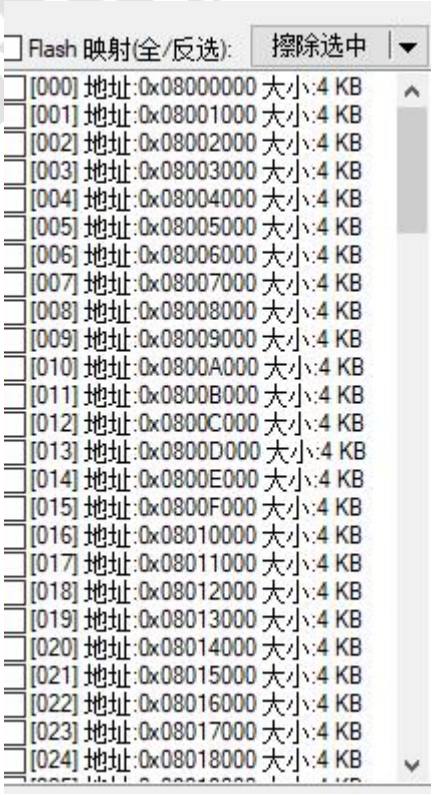


图 3.1.4.2-5 单 bank 下的 Sector 分区表

Table 7. Flash module - 512/256/128 KB dual bank organization (64 bits read width)

Flash area		Flash memory addresses	Size (bytes)	Name
Main memory (512/256/128 KB)	Bank 1 (256/128/64 KB)	0x0800 0000 - 0x0800 07FF	2 K	Page 0
		0x0800 0800 - 0x0800 0FFF	2 K	Page 1
		0x0800 1000 - 0x0800 17FF	2 K	Page 2
		0x0800 1800 - 0x0800 1FFF	2 K	Page 3
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		0x0803 F000 - 0x0803 FFFF	2 K	Page 127
	Bank 2 (256/128/64 KB)	0x0804 0000 - 0x0804 07FF	2 K	Page 0
		0x0804 0800 - 0x0804 0FFF	2 K	Page 1
		0x0804 1000 - 0x0804 17FF	2 K	Page 2
		0x0804 1800 - 0x0804 1FFF	2 K	Page 3
		-	-	-
		-	-	-
		-	-	-
		-	-	-
		0x0807 F000 - 0x0807 FFFF	2 K	Page 127

图 3.1.4.2-4 STM32G483xE 双 bank Data Sheet

Table 8. Flash module - 512/256/128 KB single bank organization (128 bits read width)

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory (512/256/128 KB)	0x0800 0000 - 0x0800 0FFF	4 K	Page 0
	0x0800 1000 - 0x0800 1FFF	4 K	Page 1
	0x0800 2000 - 0x0800 2FFF	4 K	Page 2
	-	-	-
	-	-	-
	-	-	-
	-	-	-
	-	-	-
	-	-	-
	-	-	-
	0x0807 F000 0x0807 FFFF	4 K	Page 127

图 3.1.4.2-4 STM32G483xE 单 bank Data Sheet

Power Writer 提供了最强大的选项字节功能，不仅支持完整的在线读写操作，自动识别部分预设功能，比如自动识别单双 Bank，自动检测关键选项比如保护位是否开启，支持镜像读写，同时在离线烧录模式下，提供四种预设的选项字节操作模式。

图 3.1.4.2-5 Power Writer 官方样片实测

3.1.4.3 Program Memory Tab 标签页

Power Writer 提供完整的 Program Memory 页面，这是以往大多数 Writer 都没有的功能，完整的显示当前芯片的整个 Flash Memory 布局，做到实时同步，如图 3.1.4.3-1 所示：

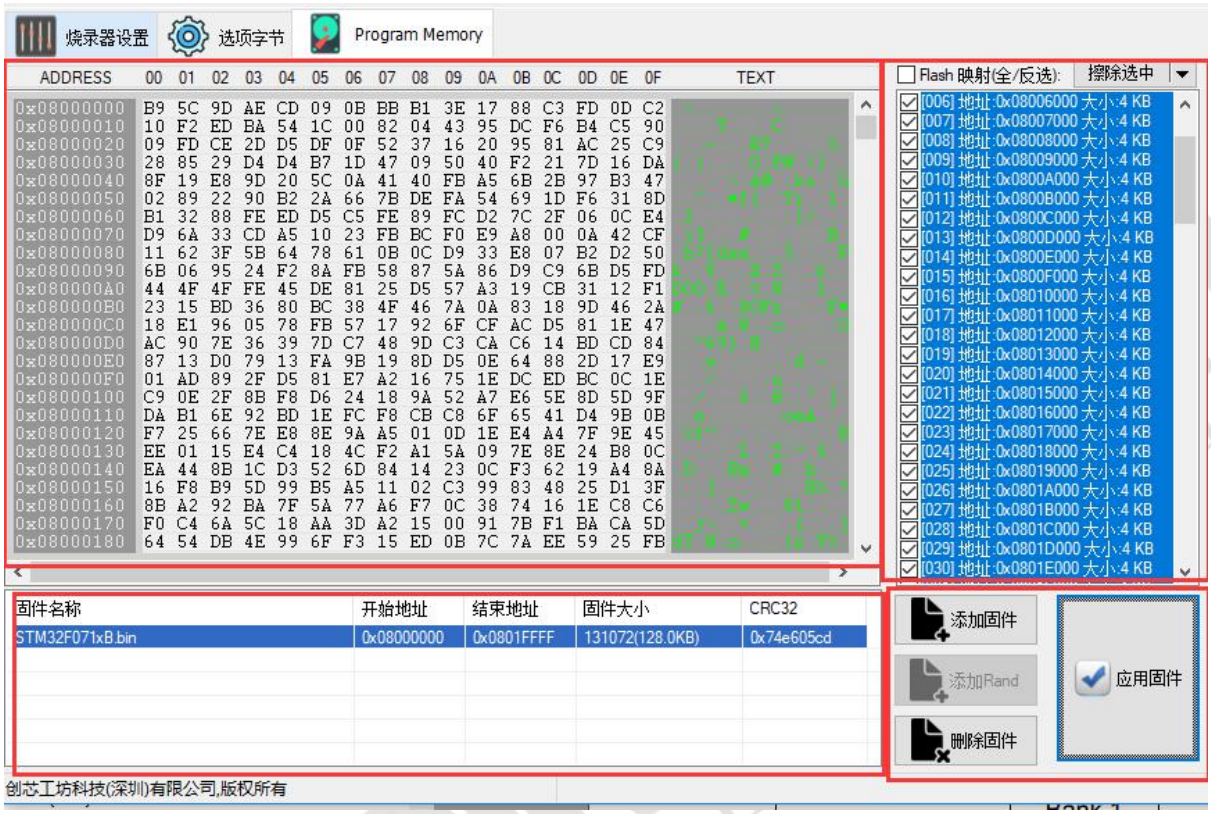


图 3.1.4.3-1 Power Writer Program Memory 标签页

Power Writer Memory 标签页分为以下几大区域：

- **Hex 数据显示编辑区**：此区域显示将要写入到芯片的数据预览，或者是实时显示从芯片读取到数据，数据显示区左侧地址栏和芯片的实际地址信息对应。右侧实时显示数据的文本模式。此区域提供快速编辑功能，通过定位光标到缓冲区，输入键盘按键即可输入任意的数据到数据缓冲区。此外提供 CTRL+C 和 CTRL+V 快速复制粘贴功能，以及提供菜单项，如图 3.1.4.3-2 所示。

图 3.1.4.3-2 Power Writer Program Memory 缓冲区菜单

复制：将选中的区段复制到剪贴板

粘贴：将选中的区段粘贴到当前光标位置

跳转到地址：Power Writer 提供快速定位地址的功能，可以无须通过右侧滚动栏寻找数据地址，如图图 3.1.4.3-3 所示，设置好地址后，点击确定按钮，将自动跳转到地址显示。

图 3.1.4.3-3 Power Writer Program Memory 跳转到地址菜单功能

- **分段固件信息显示区**：此区域实时显示 Power Writer 添加的分段固件，分段固件显示区域，支持多种格式的分段固件：文件、内存块、Mark 标记块(如随机标记块)等多种混合格式。并且在分段固件的支持上，Power Writer 没有任何的限制,此区域将显示：
  - **固件名称**：显示添加的固件名称
  - **开始地址**：显示当前分段固件的开始地址
  - **结束地址**：显示当前分段固件的结束地址
  - **固件大小**：显示当前分段固件的固件大小
  - **CRC32**：显示当前分段固件的 CRC32

- **分段固件操作区**：分段固件的操作区域，包含以下几个功能：
  - **添加固件**：Power Writer 支持多种格式的固件，S19，Hex ，Bin 格式，其他格式在后续的升级维护中，可能也会进行支持，比如 elf，axf 等格式，格式支持如图 3.1.4.3-4 所示，选择固件后点击**打开**按钮，将看到固件的设置对话框，如图 3.1.4.3-5 所示，固件设置框，将显示当前固件的文件路径、数据大小、数据大小的 KB 数，以及 16 进制显示形式，同时显示固件的起始地址(Hex,S19 自动识别起始地址，bin 格式默认显示 Program Memory 的起始地址)，以及结束地址信息，核查起始地址无误后点击确定按钮，将在固件列表中看到添加的固件。

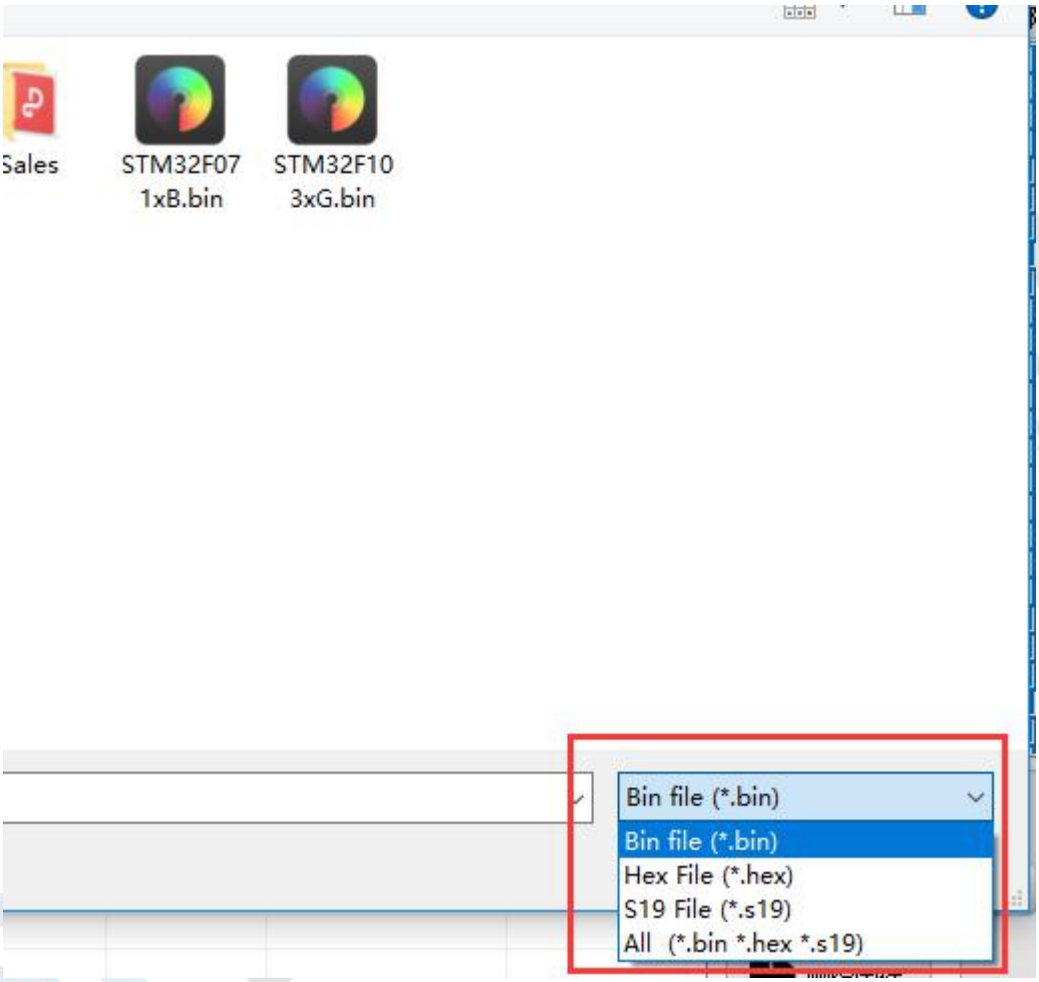


图 3.1.4.3-4 Power Writer 支持的固件格式

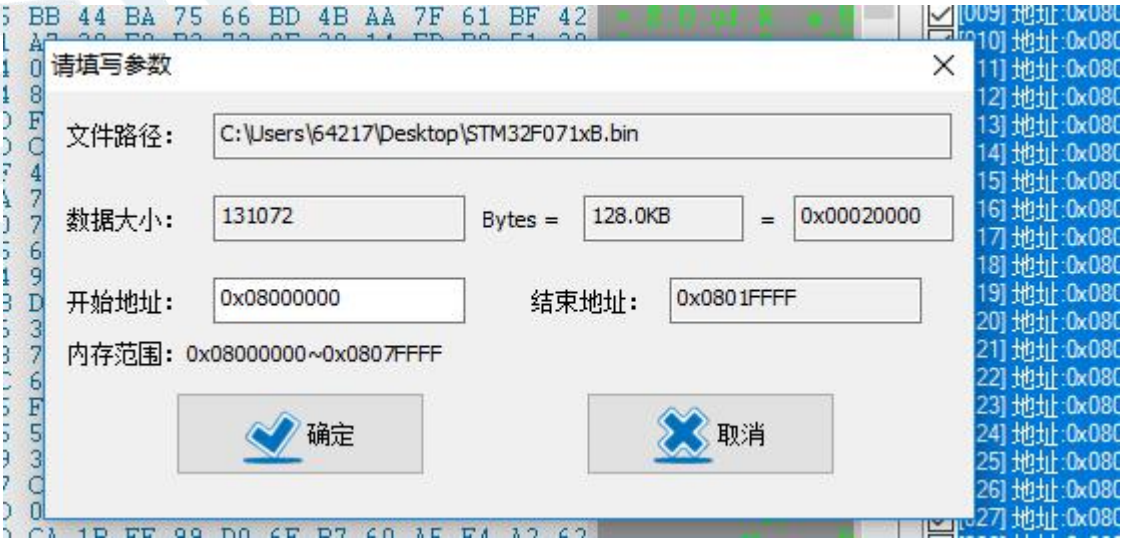


图 3.1.4.3-4 Power Writer 固件起始地址设置对口框

特别注意：

1. 固件不能出现重叠，否则将会错误对话框，如图 3.1.4.3-5 所示

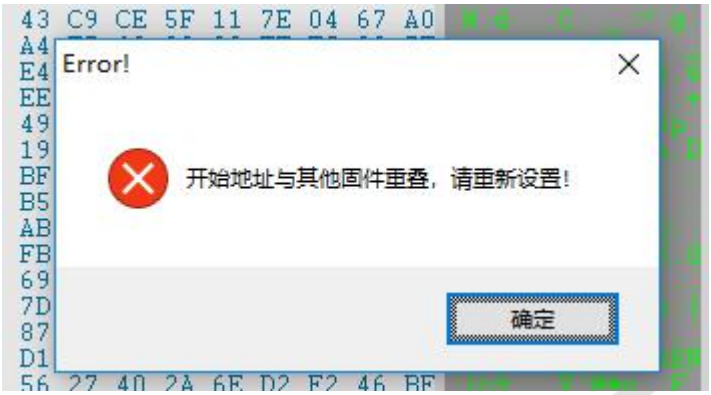


图 3.1.4.3-5 Power Writer 固件不能重叠

2. 固件不能超出 Program Memory 空间，包括起始地址和结束地址都必须在 Program Memory 所在的范围内。

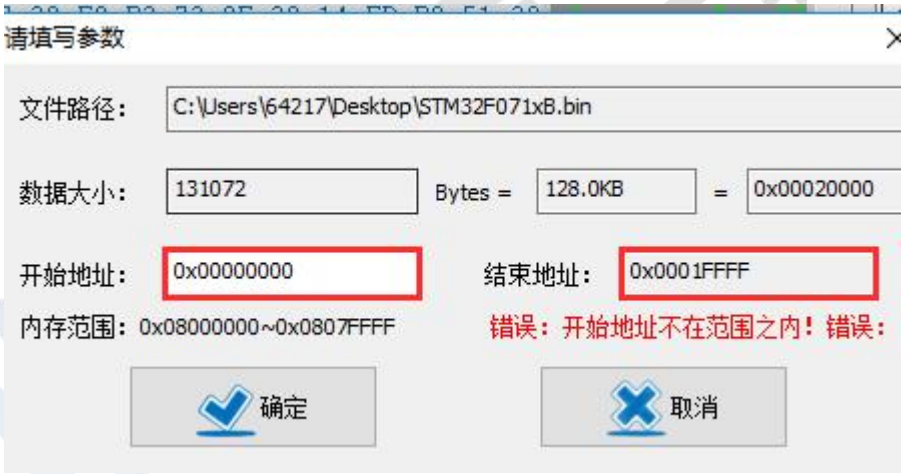


图 3.1.4.3-5 Power Writer 固件必须在 Program Memory 范围内

3. 由于部分芯片写入时必须对齐到对应的字节数，Power Writer 会尝试将用户固件回读以便补齐到需要对齐的基地址，此功能大部分时候都是可用的，但是针对部分芯片无法执行多次操作，此功能是一个实验性功能，用户在生成固件时，确保对齐固件到对应的首地址，比如芯片是 4 字节写，比如将首地址对齐到 4 字节的整数倍。如果是 32 字节写，必须对齐到 32 字节的整数倍，对齐操作依然是创芯工坊官方推介的操作，关于 Flash 对齐操作的表格请参考附录 Power Writer Flash 起始地址对齐表

- **添加 Rand**：添加随机数组标记（开发中）.
- **删除固件**：删除固件按钮可以删除选择的固件。
- **应用固件**：添加到固件列表的中固件，增删完成后，点击应用固件，Power Writer 将固件刷新到缓冲区实时显示，同时会选中当前固件对应的扇区表。

■ **Program Memory 扇区表**：扇区表是 Power Writer 独有的功能，用户可以直观地从扇区表看到芯片的 Flash 结构，并直观地显示扇区表的索引、地址、扇区大小信息，如图 3.1.4.3-6 所示。

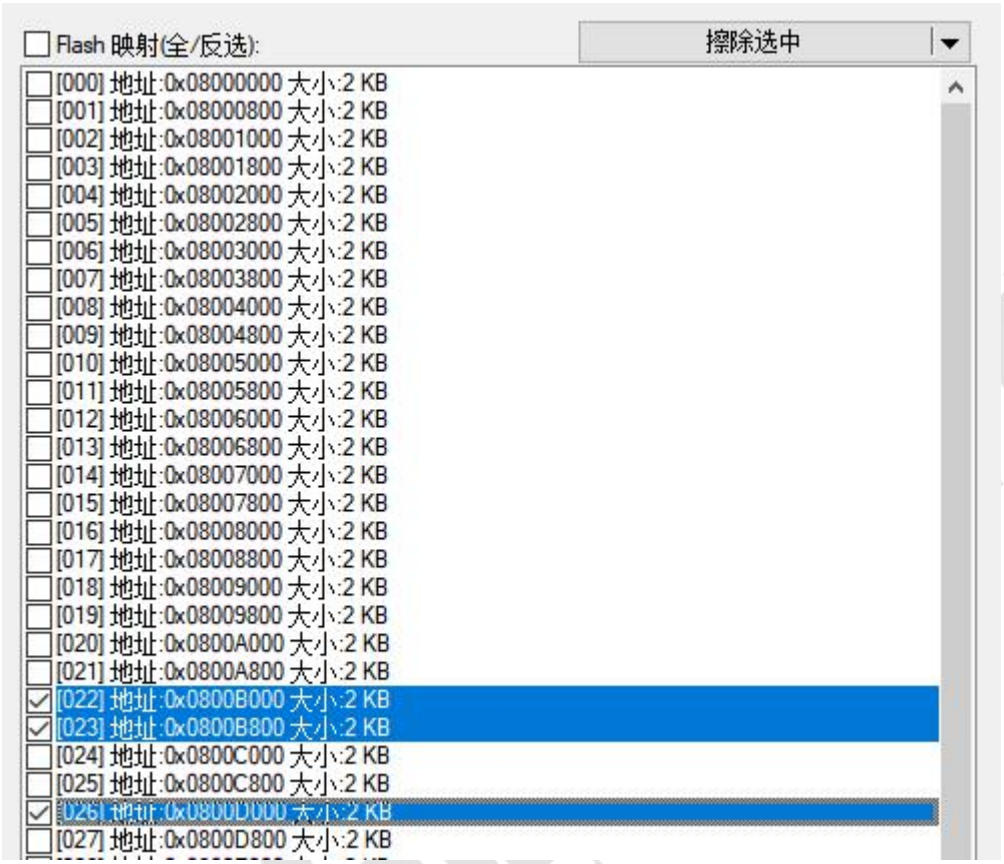


图 3.1.4.3-6 Power Writer 扇区表

Power Writer 的扇区表功能除了直观地展示给用户之外，此外额外提供了快速定位，随机填充选中扇区，擦除选中扇区功能。如下图 3.1.4.3-6 所示

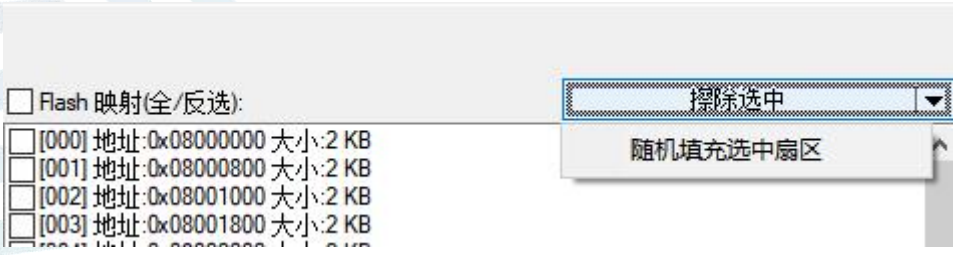


图 3.1.4.3-6 Power Writer 扇区在线操作功能

擦除选中：用户可以自由选择需要擦除的扇区，然后点击**擦除选中，**Power Writer 将对用户选中的扇区进行擦除。

随机填充：随机填充功能将随机数据填充到用户选中的扇区，常用于测试、填充 Flash 的空白数据，用于隐藏自己的真实数据等操作，用户可以自由发挥。

3.1.4.4 EEPROM Tab 标签页

部分芯片有 EEPROM,如果选择有 EEPROM 的芯片，会显示弹出 EEPROM TAB 页，以供用户导入 EEPROM 数据。

注：EEPROM Tab 目前处于开发之中，请留意 Power Writer 升级信息..

3.1.4.5 OTP Memory Tab 标签页

部分芯片有 OTP,如果选择有 OTP 的芯片，会显示 OTP TAB 页，以供用户导入 OTP 数据。
注：OTP Memory Tab 目前处于开发之中，请留意 Power Writer 升级信息..

3.1.5 Power Writer 状态栏

Power Writer 状态栏左侧显示 Power Writer 的开发方：创芯工坊科技（深圳）有限公司。右侧显示 Power Writer 项目路径信息。如图 3.1.5 所示。

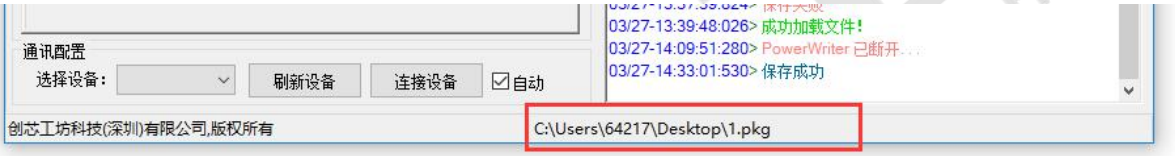


图 3.1.5 状态栏显示 Power Writer 保存项目路径

3.2ICWKEY 应用软件概览

创芯工坊作为专业的互联网+ 的新型创新型企业，一直致力于优化嵌入式软硬件生成管理流程，在芯片安全研发上不断投入，开发了 Power Writer 系列安全烧录器终端产品，配合创芯工坊全球云服务，实现多端远程量产控制，为了确保乎用户数据的安全，我们开发了大量的核心安全技术，应用于 Power Writer 系列产品上，ICWKEY 就是最典型的一款专业多加密协议的授权工具。

由于受内容篇幅的限制，本节将不详细介绍 Power Writer 和 ICWKEY 配套使用时的应用流程。具体请参考 《ICWKEY 用户参考手册 RM0002.PDF》，通常和 ICWKEY 应用软件一起发布。也可以从官方站点下载：

网址：https://www.icworkshop.com

3.3ICWKEY SDK 开发指引

ICWKEY 用户开发指南请参考 《ICWKEY 用户参考手册 RM0002.PDF》，通常和 ICWKEY 应用软件一起发布。也可以从官方站点下载：

网址：https://www.icworkshop.com

3.4创芯工坊后台管理系统概览

Power Writer 虽然可以作为开发工具独立使用，但最佳的应用方案则是利用创芯工坊强大的后

台管理系统实现远程量产、授权控制。Power Writer 不仅仅是一个 Debugger、同时也是一个 Writer、同时还是一个强大的生产工具，为了方便用户快速入手 Power Writer 的生产控制模式，本章节将概要的讲解一下创芯工坊 ICWorkShop 的后台功能，更详细的后台管理系统使用方法，请阅读创芯工坊的帮助中心文档，见 https://www.icworkshop.com/article/helpCenter/91/36，本章节将引导用户注册创芯工坊账号并测试 Power Writer 的远程操作流程，主要分为以下几个章节。

- 注册创芯工坊开发者账号
- Power Writer 固件发布到创芯工坊
- 创芯工坊客户端远程烧录 Power Writer 固件。

3.4.1 注册创芯工坊开发者账号

Power Writer 在利用远程固件量产时，需要使用创芯工坊平台的功能，平台使用注册极致，固件以订单的形式发布给指定的生产方账号，生产方登录账号进行订单生产，创芯工坊的注册流程分为以下几个步骤，完成以下几个步骤就拥有了一个完整的开发者账号，如需通过平台执行收款操作，请参考平台的帮助中心或者联系我们客户服务询问。

创芯工坊注册开发者账号流程：

- 1:点击此链接进入注册流程或者在浏览器中输入：https://www.icworkshop.com/register，进入注册，显示页面如图 3.4.1-1 所示：

图 3.4.1-1 创芯工坊注册页

注册分为手机注册和邮箱注册两种形式，两种形式流程是一样的，**我们推介使用手机进行注册，方便快捷，但笔者由于手机号已经注册了，这里使用邮箱做一个示范。**比如，这里输入笔者的测试邮箱 642175216@qq.com,填写我的账号密码，xxxxxxx,

然后点击获取验证码，如图 3.4.1-2 所示：

图 3.4.1-2 邮箱注册创芯工坊

接下来登录邮箱获取验证码，如图 3.4.1-3 所示，将验证码填写到注册页的邮箱验证码输入框，然后点击注册按钮，即可完整账号注册功能，注册完账号后将自动进入到后台控制页面，显示是否设置用户名，这一步可以取消，或者设置看个人喜好。

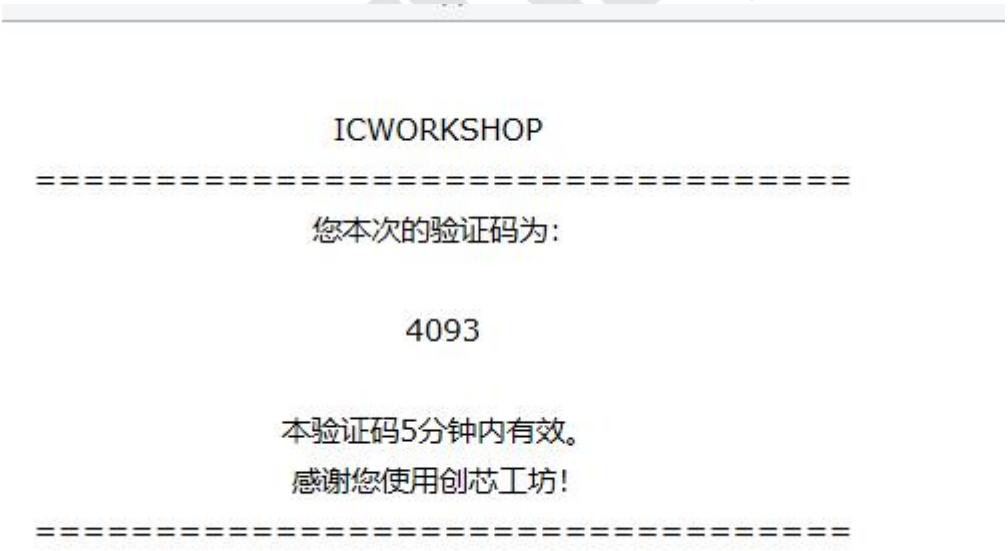


图 3.4.1-3 邮箱注册创芯工坊

图 3.4.1-4 创芯工坊注册成功

注册完账号后，即拥有了创芯工坊远程固件量产控制的能力。

3.4.2 Power Writer 固件发布到创芯工坊

经过上一部我们注册了创芯工坊的账号，这一章节，我们将登陆此账号，将固件以订单形式发布到创芯工坊的后台，登录账号后点击我是开发者->程序发布，如图 3.4.2-1 所示，

图 3.4.2-1 进入固件发布流程

如图 3.4.2-2 所示，发布 Power Writer 固件的流程如下：

- **填写项目名称：**项目名称是当前提交固件的项目名称，用户可以按照命名习惯来命名
- **填写订单发布单价：**创芯工坊的后台作为生产管理后台，有完整的订单发布流程，生产管理后台，开发方和生产方以卖家或者买家的角色进行在线交易，这里的单价是指每烧录一次芯片买家需要付费给卖家(开发方)的价格，这里可以勾选为赠送，赠送之后，将直接将当前固件发送指定的次数到买家账户上。
- **填写库存：**库存表示固件的上次烧录次数总数是多少，如果是赠送样品，则是赠送样品的数量
- **填写买方账户：**指定接收固件购买信息的账户或者是接收赠送样品的账户，可以为自己的

账户，用于实现异地远程烧录。

- **芯片信息：**这里需要需要选择芯片品牌，型号，系列，由于创芯工坊对部分芯片做了多家烧录器支持，或者是一个烧录器支持多家芯片的烧录，所以根据你选择的芯片，创芯工坊将列举出支持的烧录器。
- **填写烧录参数：**每一台烧录器都有不同的服务端参数类型，这一部分只有当选择完烧录器之后才可以看到，通常 ARM 系列的芯片都可以找到 Power Writer 的烧录器支持，如果找不到，Power Writer 对其支持仍然在适配过程中。

Power Writer 的参数选择界面如下图 3.4.2-3 所示 ，其中，上传固件格式为 **pkg 格式，此格式是 Power Writer 项目文件，**编程选项中需要填写 pkg 项目文件的密码，否则 PowerWriter 无法烧录 pkg 文件，另外一个选项是编程模式，Power Writer 的远程下载模式分为：

在线模式(创芯工坊授权服务器)：针对 Power Writer 的在线授权应用，创芯工坊内建了非对称加密算法的 ECDSA 电子签名算法，用户可以在授权中心添加项目，即可在 Power Writer 发布固件时看到授权的算法，创芯工坊内建授权算法的 SDK 参考 ICWKEY 的开发手册，授权原理同 ICWKEY 一致，只是针对某些特定客户，安全性要求比较高的芯片，以及需要管理后台授权数据的应用案例，设置页界面显示如图 3.4.2-1 所示

图 3.4.2-1 Power Writer 在线授权(创芯工坊内建)模式

在线模式(创芯工坊授权服务器)：创芯工坊同时提供授权服务器的快速搭建模板，针对用

户想搭建自己的授权服务器，可以基于创芯工坊的 SDK 做二次开发。然后在下图 3.4.2-3 所示的界面填写 自有服务器的 **授权服务器地址** 和 **授权密钥，**授权服务器的 API 格式请参考授权服务器二次开发帮助文档。

图 3.4.2-3 Power Writer 在线模式(自有授权服务器)

离线线模式：Power Writer 在离线模式下将不使用在线授权方法，而是通过 Power Writer 内置的离线功能，进行固件烧录。此模式下只需要填写 pkg 项目的密码，如图 3.4.2-4 所示。

图 3.4.2-4 Power Writer 离线模式

图 3.4.2-3 Power Writer 后台参数选择界面

当用户填写好量产时的参数数据，如果有附件，比如生产资料，说明手册等附加资料，可通过上传附件的功能上传，附件将以生产订单的定时同步到固件接收方账户。

此外，一般描述性信息通过在输入框添加描述。

一切准备就绪之后，点击 确认发布 按钮，将固件发布到生产方账户中。

图 3.4.2-3 附加生产资料提交

图 3.4.2-2 固件发布流程概览

3.4.3 创芯工坊客户端远程烧录 Power Writer 固件

经过前面的两个步骤，现在 Power Writer 即可支持远程芯片量产，量产流程相对来说比较简单，分为以下几个步骤。

- **安装创芯工坊客户端软件：**https://www.icworkshop.com/user/clientDownload，安装时一路点下一步即可，安装界面如图 3.4.3.-1 所示。

图 3.4.3.-1 创芯工坊安装界面

- **登录到创芯工坊：**当安装好创新工坊客户端软件直接，从桌面点击图标进入登录页面，如图 3.4.3-1 所示，登录成功之后，将看到赠送或者购买的固件，如图 3.4.3-2 所示。

图 3.4.3.-2 登录到创芯工坊

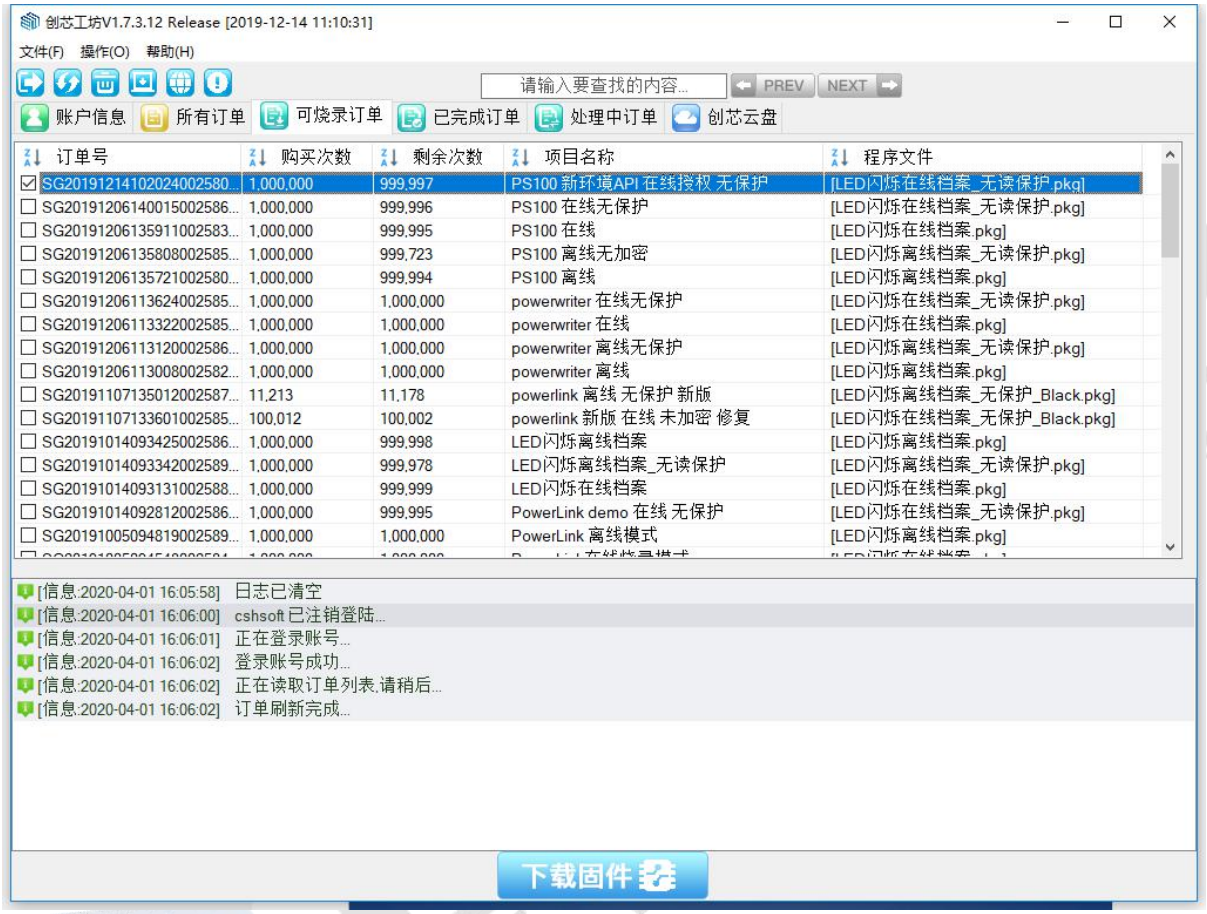


图 3.4.3.-3 创芯工坊客户端已获得授权的订单列表

- **Power Writer 固件的烧录：**Power Writer 固件的远程量产通过选择 **可烧录订单** 已获得授权的订单，然后点击下载固件按钮。将会弹出本次烧录的配置确认信息，如图 3.4.3-4 所示：

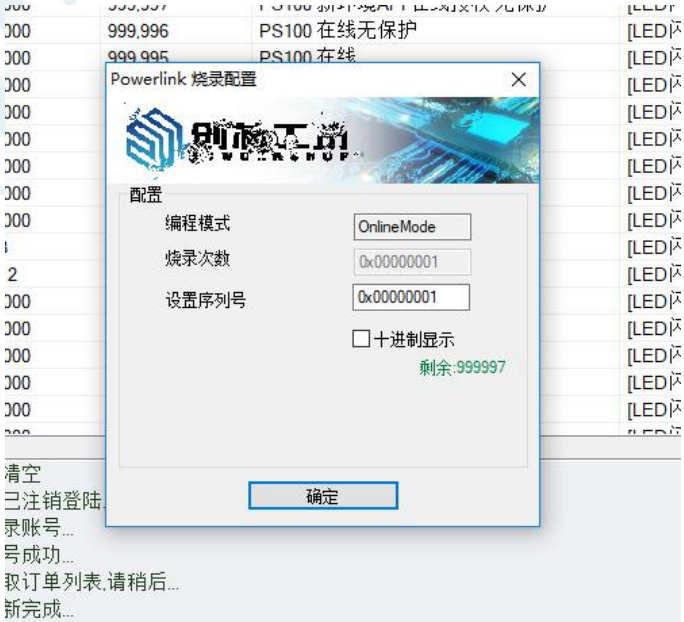


图 3.4.3.-4 创芯工坊对 Power Writer 的量产最终配置页面

当所有的设置检查无误之后，点击确定按钮，即可进入远程固件量产流程，烧录成功或者

失败，都将在创芯工坊的日志栏看到详细的日志。

注：本节的内容只是创芯工坊应用的一个快速入门指南，随着客户端软件版本的升级，以及服务端的升级，其中的信息可能会变更，详细参考请前往创芯工坊官网获取详细的帮助说明。

3.5创芯工坊 License Server 概览

Power Writer 除了内置随机矩阵 Matrix 算法、增强型 ICWKEY 授权工具之外、同时开发一套在线授权的 License Server，License Server 支持二次开发，默认采用 ECDSA 电子签名算法。开发的 SDK 同 ICWKEY 的开发 SDK，创芯工坊为了降低用户的使用门槛，提供更便捷的服务。将 License Server 作为一个授权组件集成到了创芯工坊的后台中，进入授权中的流程为：登录创芯工坊->进入我是开发者专栏->左侧点击授权中心，如图 3.5-1 所示。

图 3.5-1 License Server 在创芯工坊后台中的入口

3.5.3 创芯工坊内建 License Server 指南

请参考：https://www.icworkshop.com/article/helpCenter/93/36

3.5. 4 基于创芯工坊 License Server SDK 自建授权服务器指南

请参考：https://www.icworkshop.com/article/helpCenter/92/36

4 ： Power Writer 应用指南

本节将介绍 Power Writer 的快速应用示范，以方便用户快速入手 Power Writer 应用，Power Writer 应用部分，主要分为三种类型：**PowerWriter 开发者应用、PowerWriter 标准烧录应用和 PowerWriter 创芯工坊应用三种类别**，接下来的步骤将概要的介绍着三种主要应用模式，引领用户

快速入手 Power Writer 应用。

4.1Power Writer 开发者应用

Power Writer 开发者应用描述了 Power Writer 提供给开发人员比较可能用到的功能，提供给开发者的常用功能，主要包含两种常用的应用类型：作为 Debugger 使用、作为在线烧录器使用。

4.1.1 Power Writer 开发者功能之 Debugger

用户不单是将 Power Writer 是作为烧录器使用，与此同时，Power Writer 提供了完整的 ARM 芯片 Debugger 功能(STM8 的 Debugger 功能正在开发)，极大地方便了开发者，同时不同于市面上的 Debugger 只支持单一品牌芯片，如 STLINK、MMLINK、GDLINK、NU-LINK 等官方发布产品只支持单一芯片，要么就是支持全品牌的 Debugger 如 J-Link 等产品，但是仍然暴露出它的不足、只是一个 Debugger 使用，而没有离线量产，授权控制功能，要么就是单纯的 Flasher，比如 Flasher ARM ,但针对开发者和用户来说成本又显得过于高昂，创芯工坊的 Power Writer 系列产品就提供了全功能特性的同时，同时不断的适配不同厂家极大的方便用户。

下面以 Keil(MDK) 应用为例，描述了 Power Writer 作为 Debugger 在 ARM 开发中的应用方法，如图 4.1.1 所示：

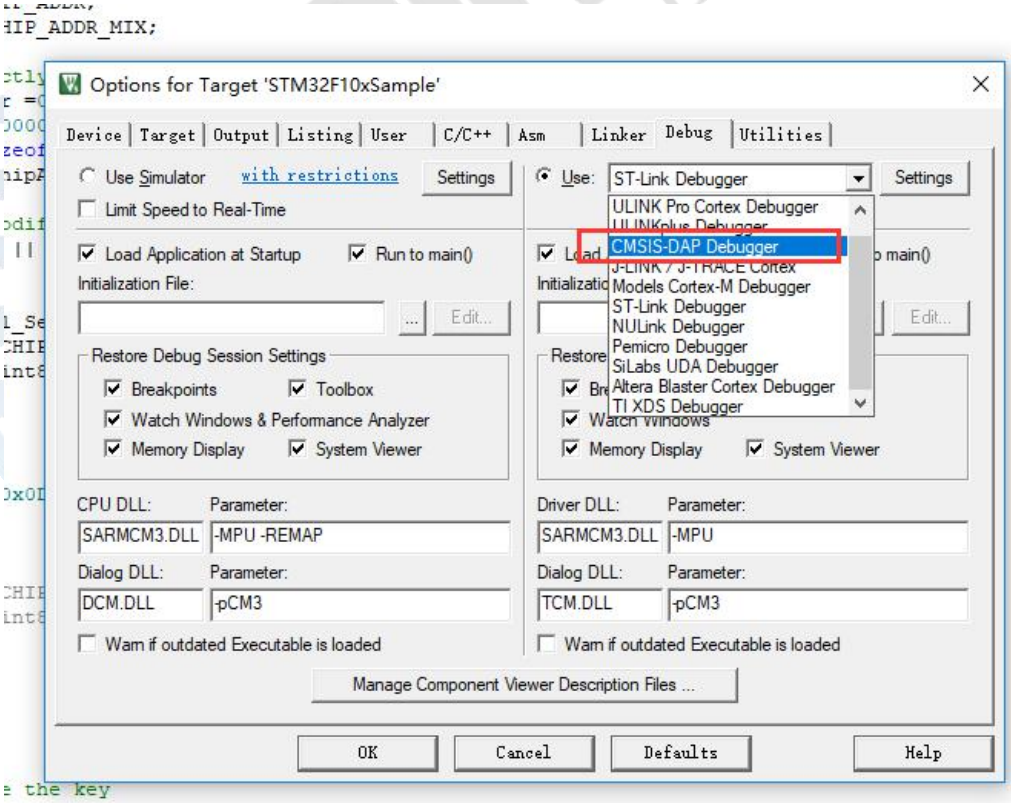


图 4.1.1 Power Writer 作为 Debugger 时选择 CMSIS-DAP Debugger

接下来点击设置页按钮，进入 Debugger 的设置页，如图 4.1.1-2 所示，从左侧列表选择已经连接好的 CMSIS-DAP ，选择 Port 为 SW 模式，选择 CLOCK，最高可以选 10M，其余保留默认。

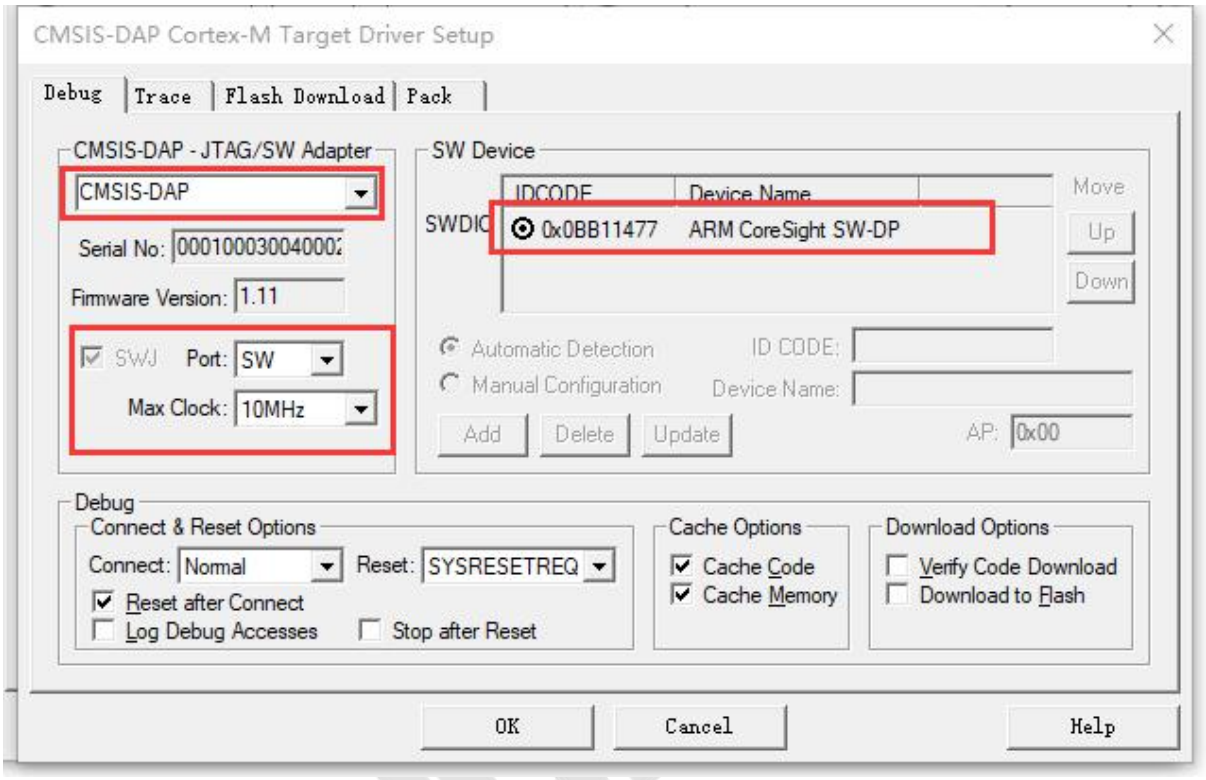


图 4.1.1-2 Power Writer Debugger 设置项

最后一步，需要添加 Debugger 所需的 Flash Algorithm,点击 Flash Download 标签页进入到下载配置页面，点击 ADD 按钮从列表中选择对应芯片的 Flash Algorithm，如图 4.1.1-3 所示：

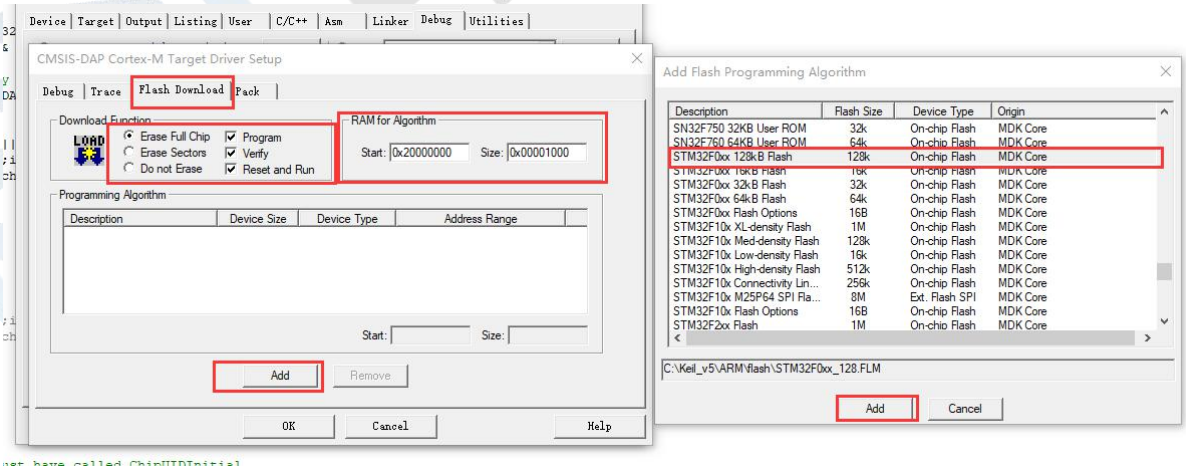


图 4.1.1-3 Power Writer Debugger 添加 Flash Algorithm

当添加好 Flash Algorithm 之后，点击确定关闭窗口，回到 MDK 主页面，点击工具栏的  按钮或者键盘的 **CTRL + F5** 即可进入 Debug 模式，如图 4.1.1-4 所示，进入 Debug 功能后，Power Writer 的 Debug 功能和 STLINK 、NULINK、GDLINK、J-LINK 功能一致。

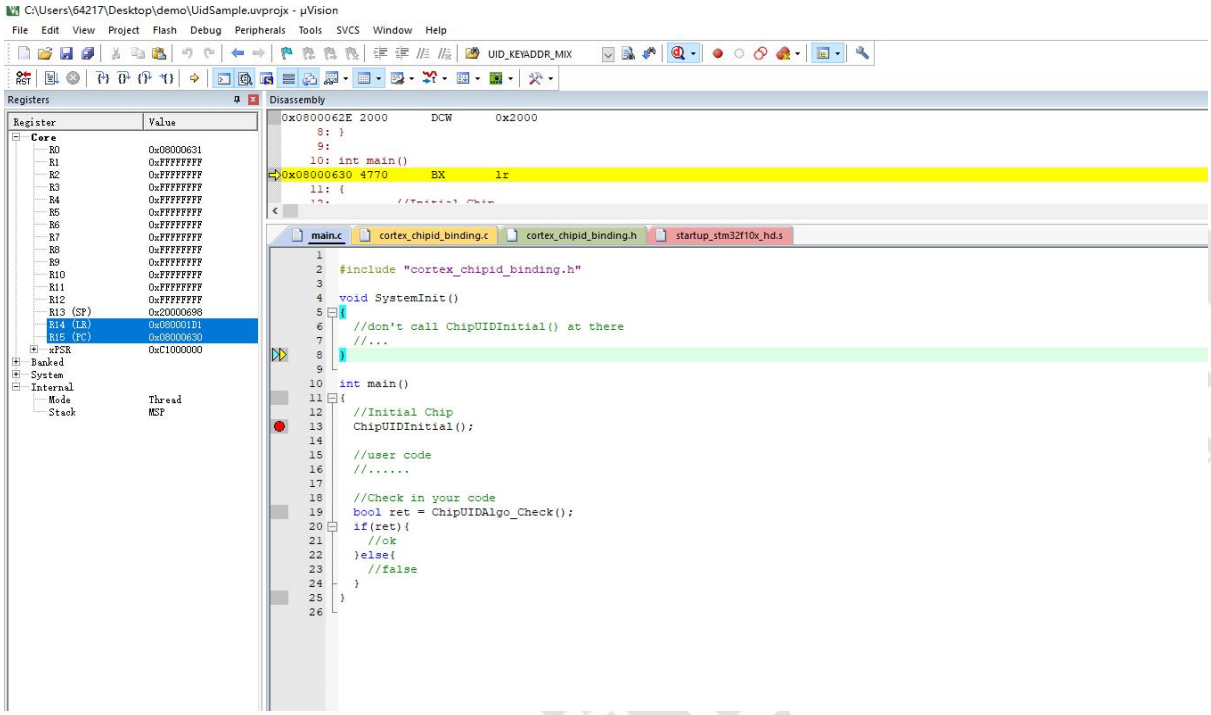


图 4.1.1-4 MDK 搭配 Power Writer 进行项目 Debug

注意:

- 其他 IDE 的应用方式和 MDK 基本一致，比如 CUBEIDE,IAR 等 IDE 工具，我们常用 MDK，就以 MDK 作为参考。
- 如果在 Flash Algorithm 找不到指定芯片的 Algorithm 项，需要从 MDK 官方下载旧版本的 设备包，下载链接见： http://www2.keil.com/mdk5/legacy 或者直接点击此链接下载：
  Cortex-M 的芯片支持包：
  https://armkeil.blob.core.windows.net/legacy/MDKCM525.EXE 和
  ARM7、ARM9、和 Cortex-R 的支持包：
  https://armkeil.blob.core.windows.net/legacy/MDK79525.EXE。

注：其他 IDE Debugger 功能需要使用 OpenOCD（GDBServer 做代理），具体可以通过百度来寻求解决方法，后续我们也将补充这方便的资料。

4.1.2 Power Writer 开发者功能之 Option Byte 读写

Power Writer 的在线操作功能都是开发者可能常用到的功能，Power Writer 的在线功能和 ST Programmer 软件类似，并且额外增加了 ST Programmer 没有涉及的其它功能，本节介绍 Power Writer 如何快速读写芯片的 Option Byte。

- **Option Byte 的读取：**

  Power Writer 支持在线读写 Option Byte，通过菜单读取选项字节和写入选项字节实现，如图 4.1.2 -1 所示，在线读取 Option Byte 将同步设置 Power Writer 应用软件端，支持有读报保护芯片的镜像读取，有一个特例是如果芯片的保护级别为

LEVEL-2，则无法读写目标芯片。

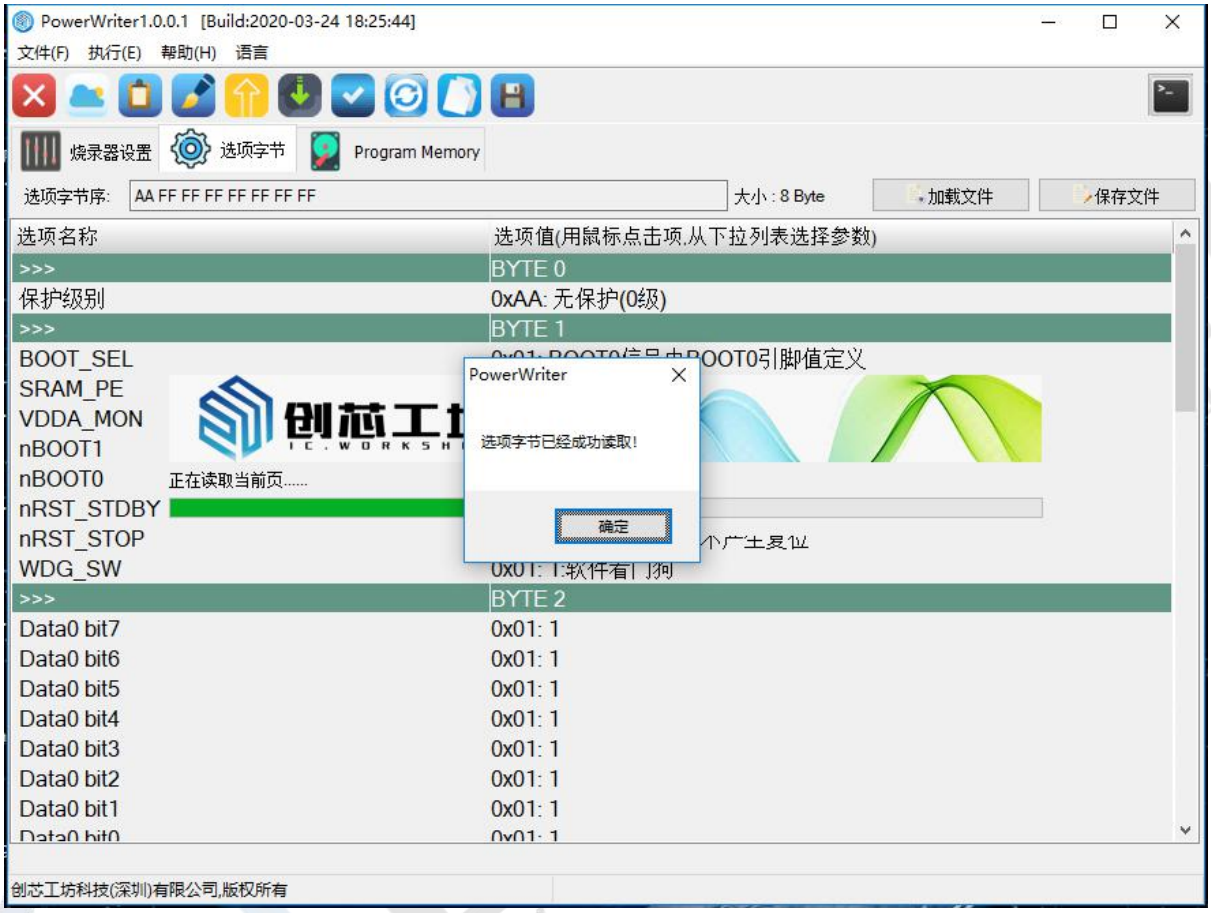


图 4.1.2 -1 Power Writer 在线读取选项字节

■ **Option Byte 的写入：**

Power Writer 支持在线读写 Option Byte，通过菜单读取选项字节和写入选项字节实现,写入选项字节时，Power Writer 内部将复位目标芯片，使其无须断电即可生效，（注：部分芯片需要接入 RESET 硬件 Pin 引脚），通过 Power Writer 更新选项字节显示结果如图 4.1.2 -2 所示：

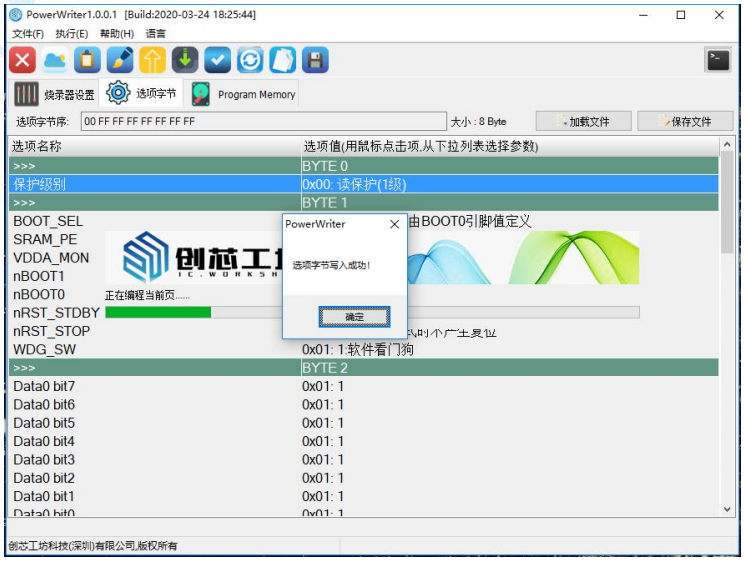


图 4.1.2 -2 Power Writer 在线更新选项字节

4.1.3 Power Writer 开发者功能之 Program Memory 操作

Power Writer 同 STLINK-V2 一样，同样支持在线读写目标芯片的 Flash，**操作前需要选择目标芯片，并连接到目标芯片，**确保芯片在线之后，可以通过菜单栏的 Program Memory 操作菜单在线读写目标芯片，如图 4.1.3-1 所示：

图 4.1.3 -1 Power Writer 在线操作 Program Memory

4.1.4 Power Writer 开发者功能之其他在线操作

Power Writer 同时提供其他开发者可能需要用到的功能，比如复位目标芯片，读取 CID，任意地址读数据等操作，这些操作都可以通过菜单栏实现，如图 4.1.4-1 所示

图 4.1.3 -1 Power Writer 在线其他操作

4.2Power Writer 标准烧录应用

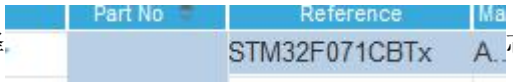
为了演示 Power Writer 的标准烧录操作，我们准备了一个全新的测试项目，然后基于此项目一步一步地演示 Power Writer 在离线烧录上的应用流程。

4.2.1 Power Writer 项目开发流程

测试项目基于 STM32F071xB 的芯片，首先我们将 STM32F071CB 的测试板连接到 Power Writer 端，并连接好 USB 线到电脑，如图 4.2.1-1 所示：

图 4.2.1-1 首先 STM32F071CB 的最小系统板和 Power Writer 连接

接下来我们基于 STM32 CubeMax，生成一个测试项目，首先打开 STM32CUBEMAX 执行新建 MCU 项目 

，选择 

芯片进入到系统配置页面，测试板子我们预留了 LED，IOPIN 为 **PC13，**为了方便测试，我们用串口和 PC13 来验证 Power Writer 的结果信息，下图是 STM32Cube Max 的 STM32F071 项目设置界面，如图 4.2.1-2/3/4 所

示：

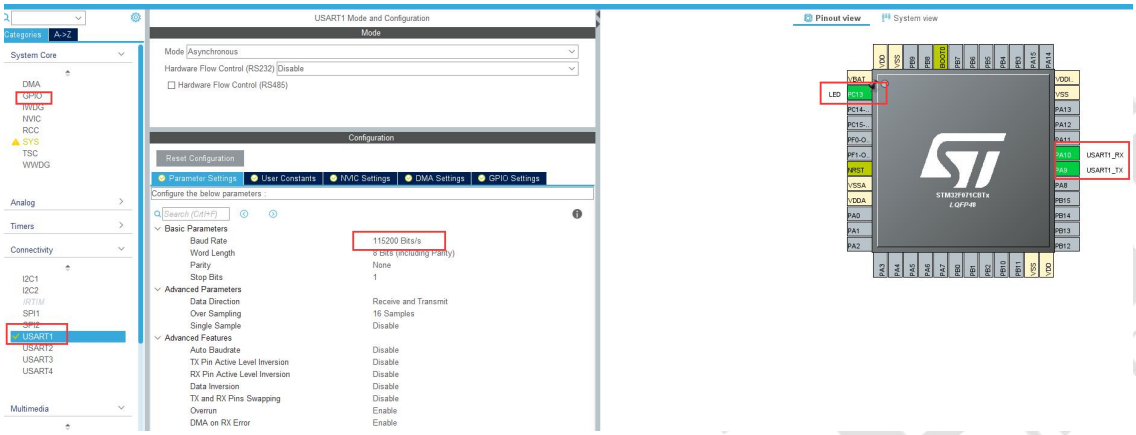


图 4.2.1-2 测试板开启 PC13 输出和 USART1 PA9/10(TX,RX)

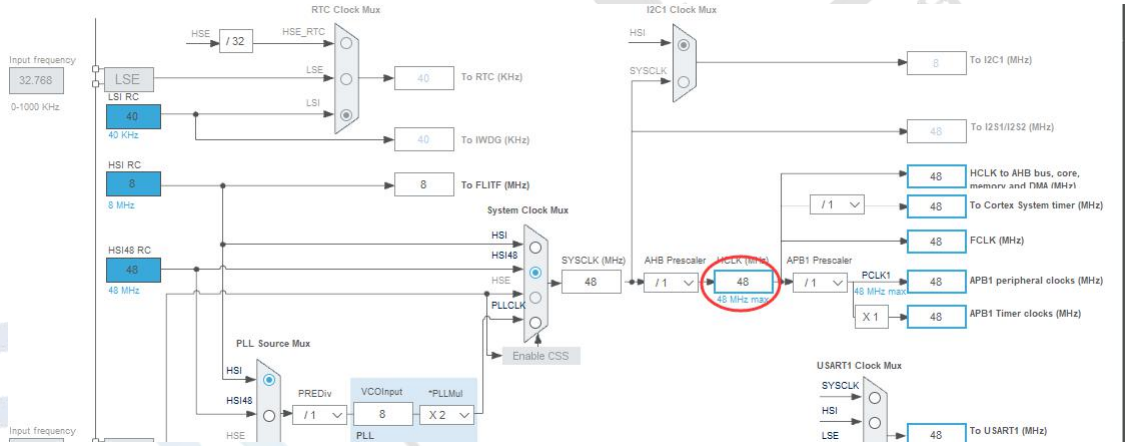


图 4.2.1-3 将时钟改为 48M 输出

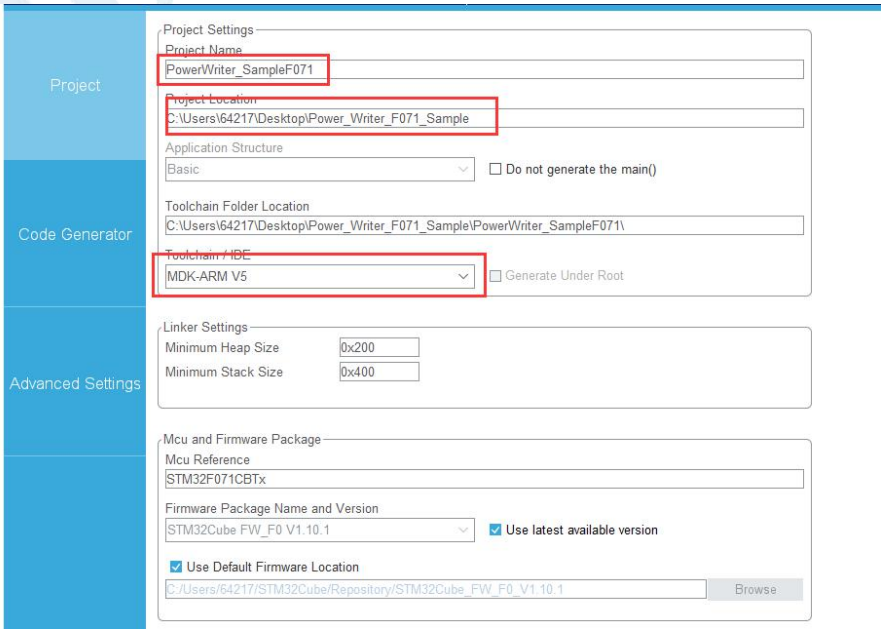


图 4.2.1-4 设置项目名称选择开发工具

当设置完成之后我们导出 MDK 项目,并设置好 Debugger ， Power Writer 同时支持

Debugger，所以在 Debug 页面直接选择 CMSIS-DAP 即可，设置好参数，如图 4.2.1-5 所示：

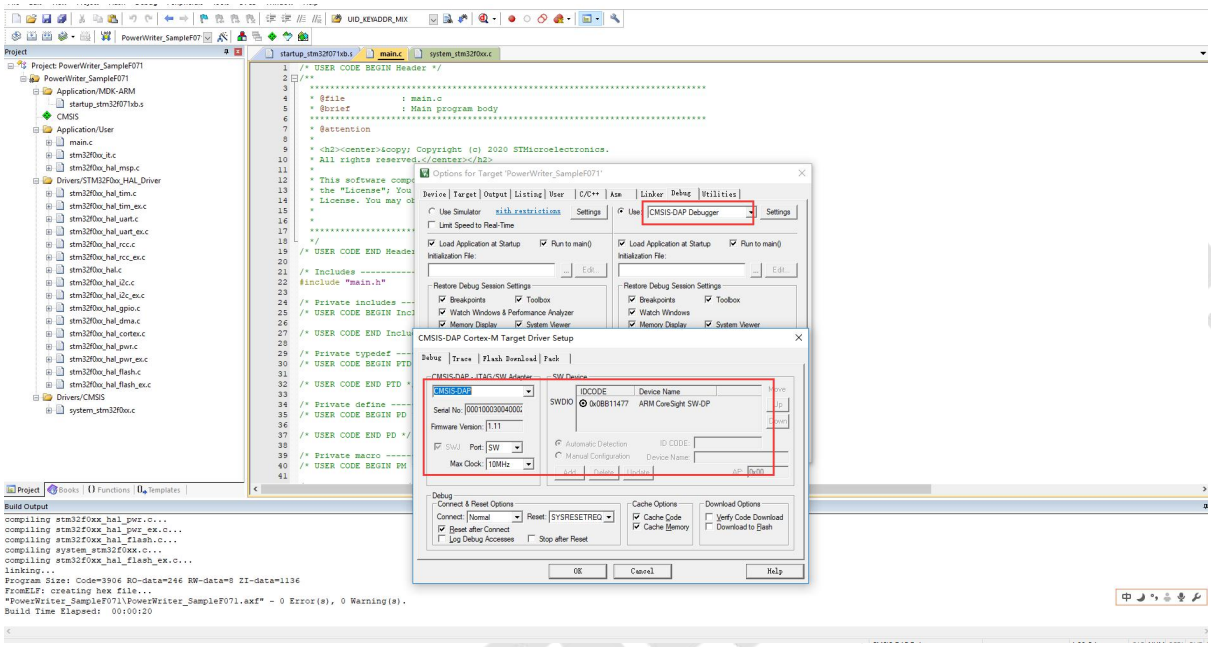


图 4.2.1-5 导出 MDK 项目并且设置 Power Writer 为 Debugger

为了方便演示，我们添加日志打印功能和 LED 显示功能，见图 4.2.1 -6 所示，用于后期信息的指示和显示，并连接好 USB 转串口工具：PA9/10(TX,RX)。

```
#ifndef _POWERWRITER_H_
#define _POWERWRITER_H_

#include <math.h>
#include <string.h>
#include <stdlib.h>

#define LOGGER_EN 1

#define LED_ON  HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);  //LED ON
#define LED_OFF HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);    //LED OFF

void powerwriter_show_title(void);

typedef enum e_logType
{
  Info,
  Debug,
  Warn,
  Excep,      //Exception
  Error,

  logTypeMax
}e_logType;

static const char * ALogType[logTypeMax] =
{
  "Info","Debug","Warn","Exception","Fatal Error"
};

#if LOGGER_EN

#define logger(log,format,...)  printf("[%s]:"format,\
  ALogType[log],\
  ##__VA_ARGS__)


#else
  #define logger(log,format,...)
#endif

#endif
```

```
//log
HAL_StatusTypeDef status;
int fputc(int ch, FILE *f)
{
    status = HAL_UART_Transmit(&huart1,(uint8_t *)&ch,1,10);
    return (ch);
}

void powerwriter_show_title(void)
{
  logger(Info,"------------------------------------------------------\r\n");
  logger(Info,"  Power Writer STM32F071CBT6 sample demo for test,\r\n");
  logger(Info,"  This Demo is a good reference sample to validate \r\n");
  logger(Info,"  the internal capabilities and details of Power Writer.\r\n");
  logger(Info,"  by:csh@icworkshop.com\r\n");
  logger(Info,"------------------------------------------------------\r\n");
}
```

图 4.2.1 -6 给项目添加 LED 操作和 日志打印函数 logger

准备妥当后，我们连接串口并启动 Debug，如果看到串口看到如下日志，说明测试项目已经准备妥当，准备接下来的测试操作。

Power Writer 支持写入序列号 SN，为此我们在项目代码中定义一个 m_serial_number 的占位符，并定义成 const 属性，以便让其内容在 Flash ROM 中，如图 4.2.1 -7 所示：

```
6
7  const uint8_t m_serial_number[4] = {0x5A,0xA5,0xA5,0x5A}; //define sn placeholder
8
```

```
7  void powerwriter_print_detail(void)
8  {
9    logger(Info,"Product serial number: 0x%08x (%d)\r\n",(*(uint32_t *)m_serial_number),(*(uint32_t *)m_serial_number));
0  }
```

图 4.2.1 -6 定义 SN 的占位符，并提供日志打印

解析来我们打开 Power Writer 应用软件，并选择 STM32F071XB 这颗芯片，开启 SN 功能，SN 的 Flash 地址获取方式，通过编译输出的 map 文件 也就是 **PowerWriter_SampleF071.map 这个文件**，通过搜索我们可以知道 SN 的存放地址为 0x080017c8,如图 4.2.1-7 所示：

PowerWriter_SampleF071.map | PowerWriter_SampleF071.lnp | PowerWriter_SampleF071_PowerWriter_SampleF071.dep

1288	UART_WaitOnFlagUntilTimeout	0x080010d9	Thumb Code	98	stm32f0xx_hal_uart
1289	__0printf$8	0x0800113d	Thumb Code	24	printf8.o(i.__0pri
1290	__1printf$8	0x0800113d	Thumb Code	0	printf8.o(i.__0pri
1291	__2printf	0x0800113d	Thumb Code	0	printf8.o(i.__0pri
1292	__scatterload_copy	0x0800115d	Thumb Code	14	handlers.o(i.__sca
1293	__scatterload_null	0x0800116b	Thumb Code	2	handlers.o(i.__sca
1294	__scatterload_zeroinit	0x0800116d	Thumb Code	14	handlers.o(i.__sca
1295	fputc	0x080015f1	Thumb Code	22	powerwriter_config
1296	main	0x08001611	Thumb Code	26	main.o(i.main)
1297	powerwriter_print_detail	0x0800162d	Thumb Code	20	powerwriter_config
1298	powerwriter_show_title	0x08001675	Thumb Code	54	powerwriter_config
1299	m_serial_number	0x080017c8	Data	4	powerwriter_config
1300	AHBPrescTable	0x080017cc	Data	16	system_stm32f0xx.o
1301	APBPrescTable	0x080017dc	Data	8	system_stm32f0xx.o
1302	Region$$Table$$Base	0x08001814	Number	0	anon$$obj.o(Region
1303	Region$$Table$$Limit	0x08001834	Number	0	anon$$obj.o(Region

图 4.2.1-7 通过 Map 文件获取 sn 的存放地址

MDK 开启 Map 文件的方式如图 4.2.1-a 所示，其他的编译器也有类似的功能来查看 Map 文件，查看 Map 文件对于我们开发固件是一个非常有用的功能，可以快速获取我们需要的信息，或者查找

项目的 bug。

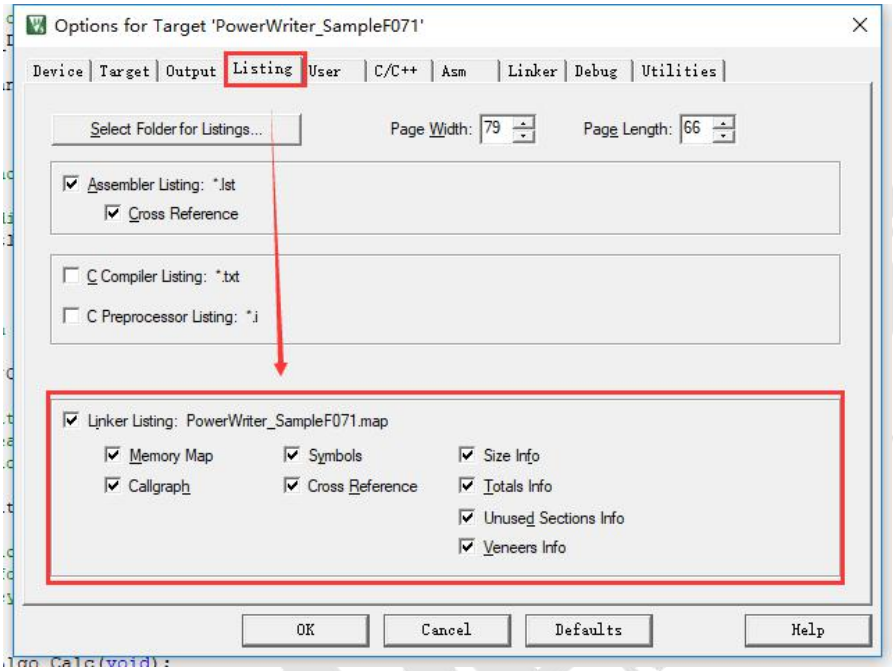


图 4.2.1-a MDK 中开启 Map 文件的操作方式

我们来验证一下 sn 在 Flash 中的地址是不是 0x080017C8,我们通过 Power Writer 添加项目的 hex 到 Program Memory 区域，通过检查 SN 确实在 0x080017C8 的位置，如图 4.2.1-8 所示：

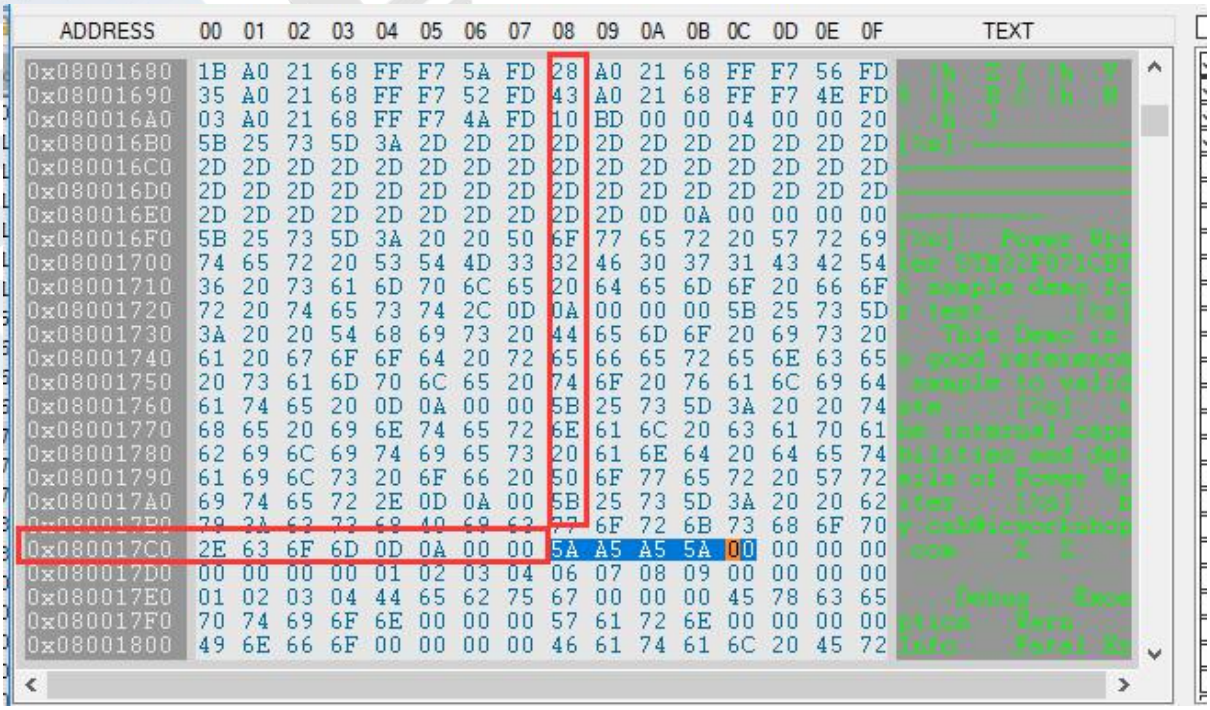


图 4.2.1-7 SN 地址验证

接下来我们为测试项目添加一个烧录器内置 Matrix 随机矩阵验证功能，我们在 Power Writer 应用软件界面 进入 UID 加密设置页，选择自带内置离线授权，填写密钥地址为：0x08002000 的位置，用户根据实际的应用情况调整，并点击 Matrix 编码，随机生成一段绑定

代码，然后导出源码，如图 4.2.1-8 所示：

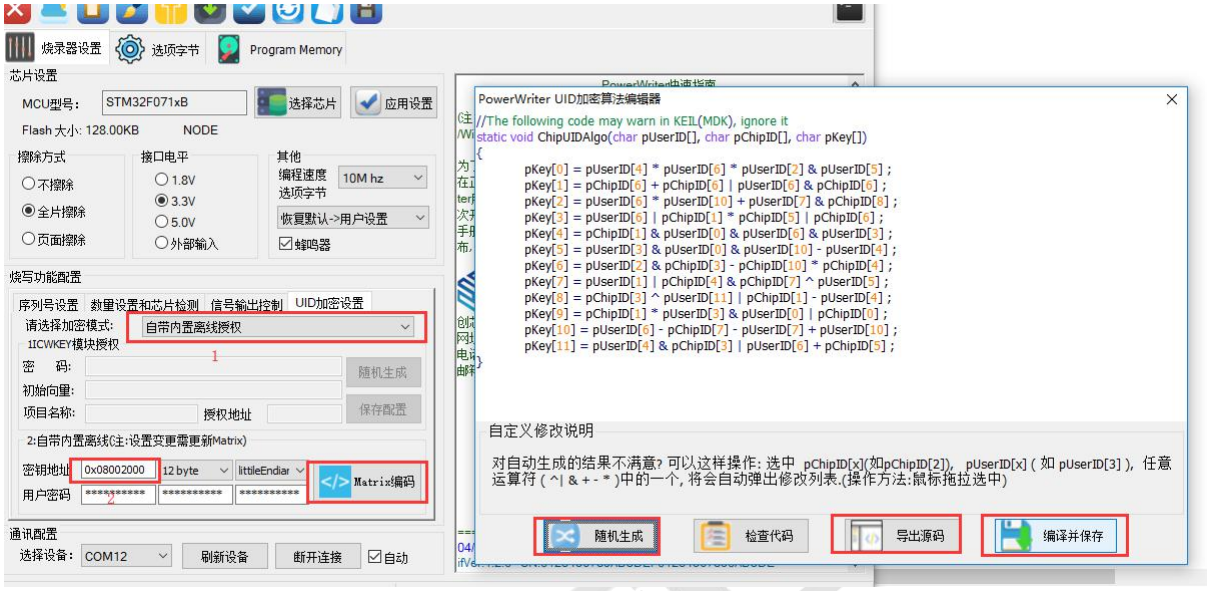


图 4.2.1-7 Power Writer 导出 Matrix 加密源码

将导出的源码加入 PowerWriter_SampleF071 项目中，如图 4.2.1-7 所示：

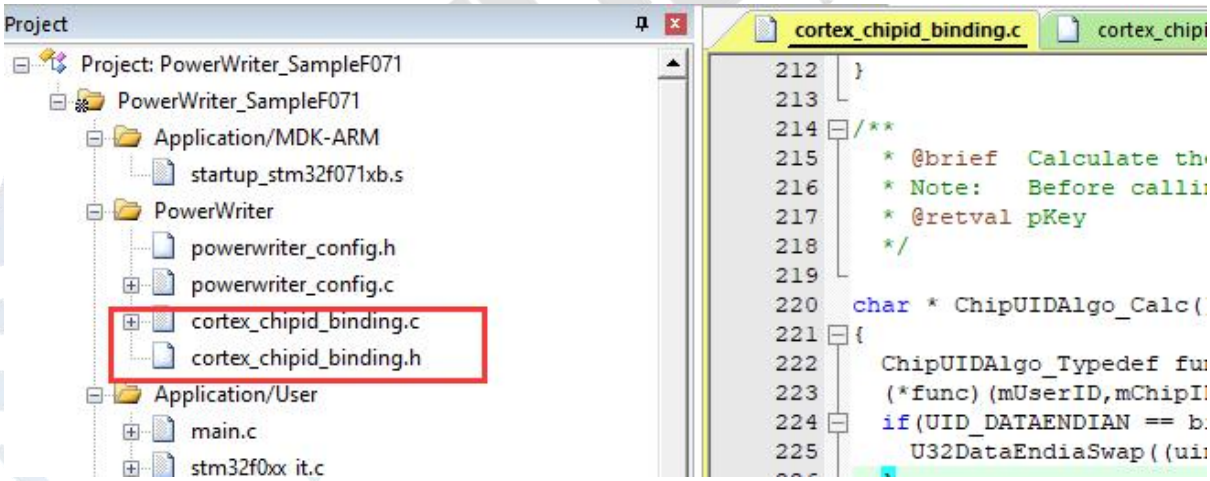


图 4.2.1-7 在测试项目中加入烧录器内置 Matrix 代码

完成以上步骤之后，我们在项目中加入测试代码，通过串口打印 SN 和 验证内置 Matrix 算法是否有效，如图 4.2.1-7 所示，此时序列号和 Matrix 算法都是无效的，因为没有经过 Power Writer 的离线量产。

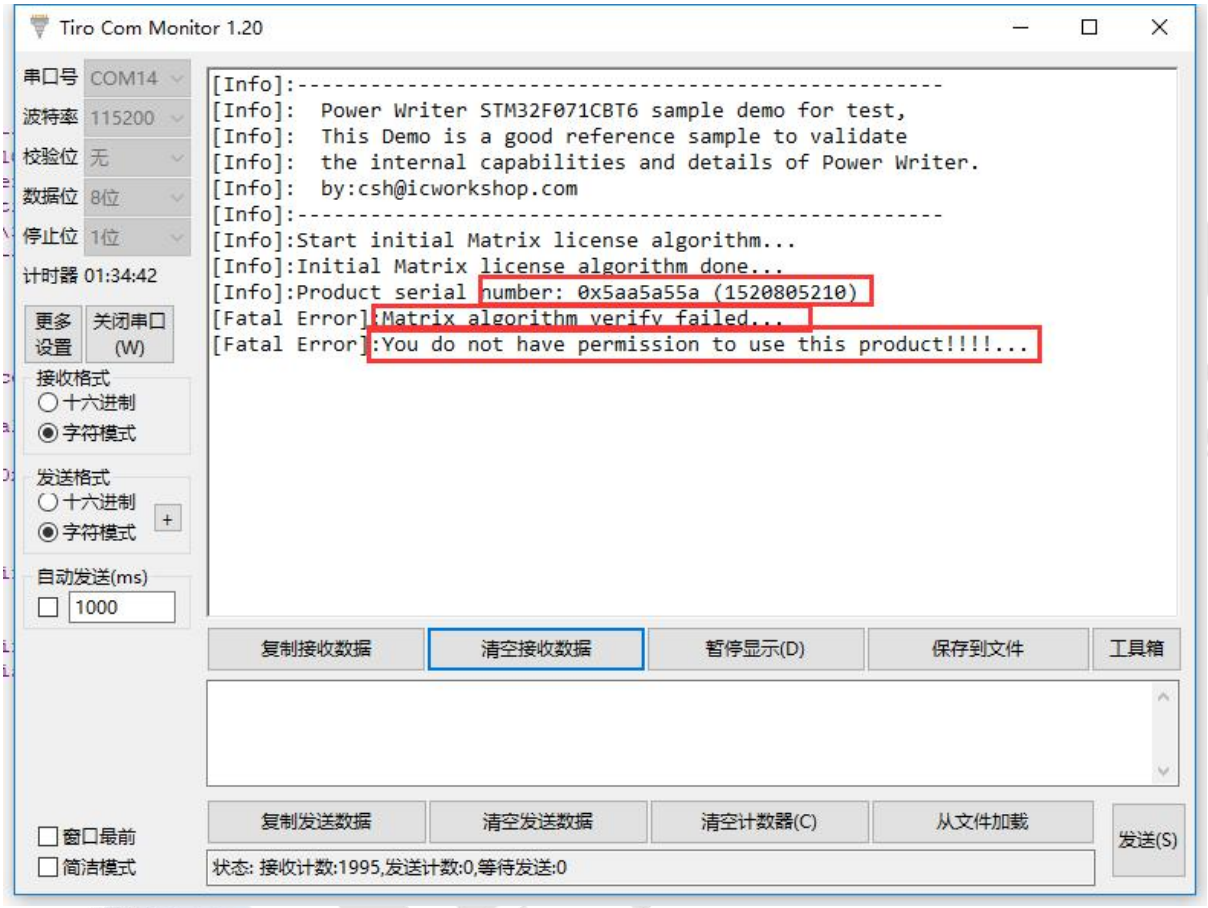


图 4.2.1-7 测试 Demo 默认的日志打印

4.2.2 Power Writer 标准离线固件打包

经过 4.2.1 节的准备，我们准备了一个测试 Demo，并且提供了日志打印功能，方便为接下来的演示，在这里，我们需要新建一个 Power Writer 的离线项目，第一页的设置截图参考图 4.2.2-1 所示，我们进行了如下设置：

- 设置为全片擦除
- 设置接口电平为 3.3V
- 设置编程速度为 10MHZ
- 设置 Option Byte 为 恢复默认->写入用户定义。
- 开启蜂鸣器
- 设置序列号初始值为 0，步进为 0x10,序列号地址为 0x08001b64（**查看 MDK 的 map 文件**）

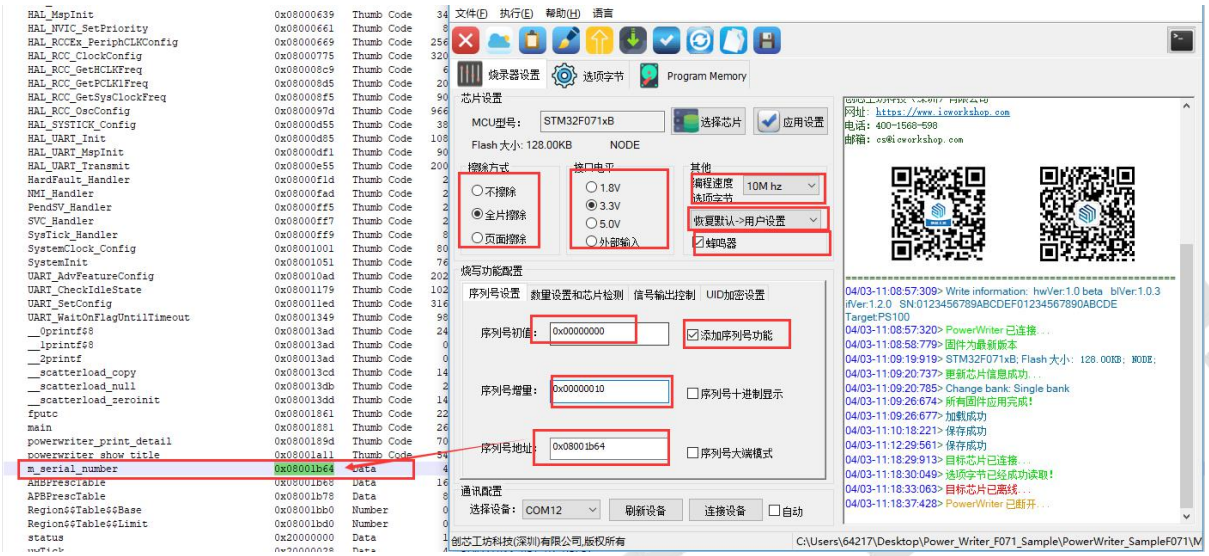


图 4.2.2-1 Power Writer 设置第一页

接下来我们设置烧录次数为 10 次，见图 4.2.2-2 所示：

图 4.2.2-2 Power Writer 设置第二页

同时设置信号输出控制中的 烧录完启动芯片，设置复位方式为输出复位后关闭(烧录完毕之后的动作)，见图图 4.2.2-3 所示：

图 4.2.2-3 Power Writer 设置第三页

UID 我们采用 Power Writer 内置的 Matrix 算法，项目基于 Power Writer 导出的源码编译的，所以信息是同步的，参考图 4.2.2.-3 所示：

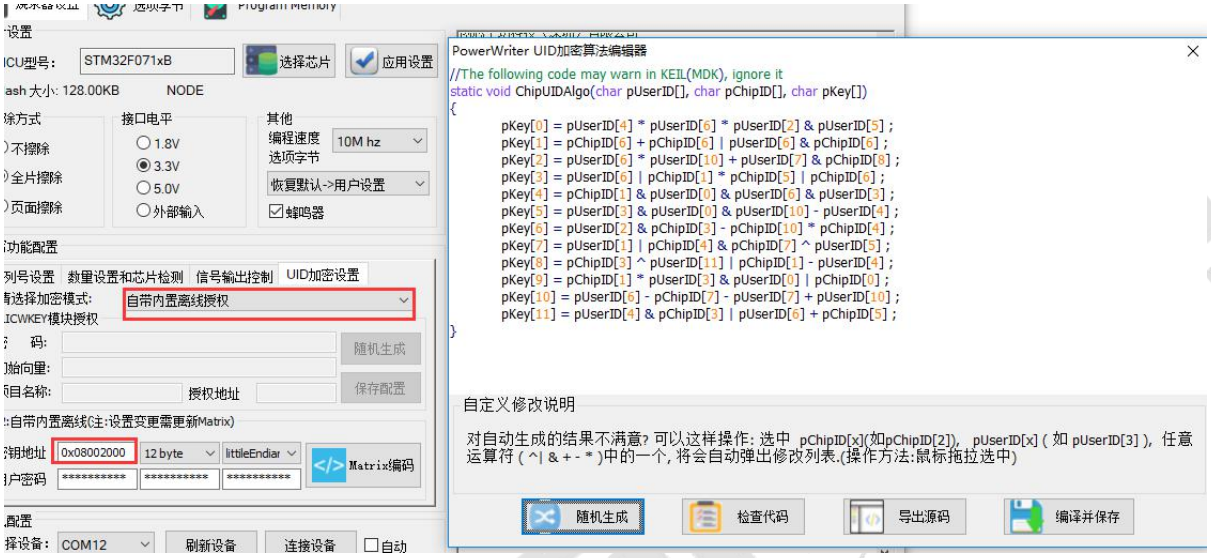


图 4.2.2-3 Power Writer 设置第四页

选项字节的设定，我们不开保护，方便验证数据，见图 4.2.2-4 所示：

图 4.2.2-3 Power Writer 选项字节不开保护

接下来我们添加 Demo 编译出来的固件到 Program Memory 中，添加后如图 4.2.2-4 所示

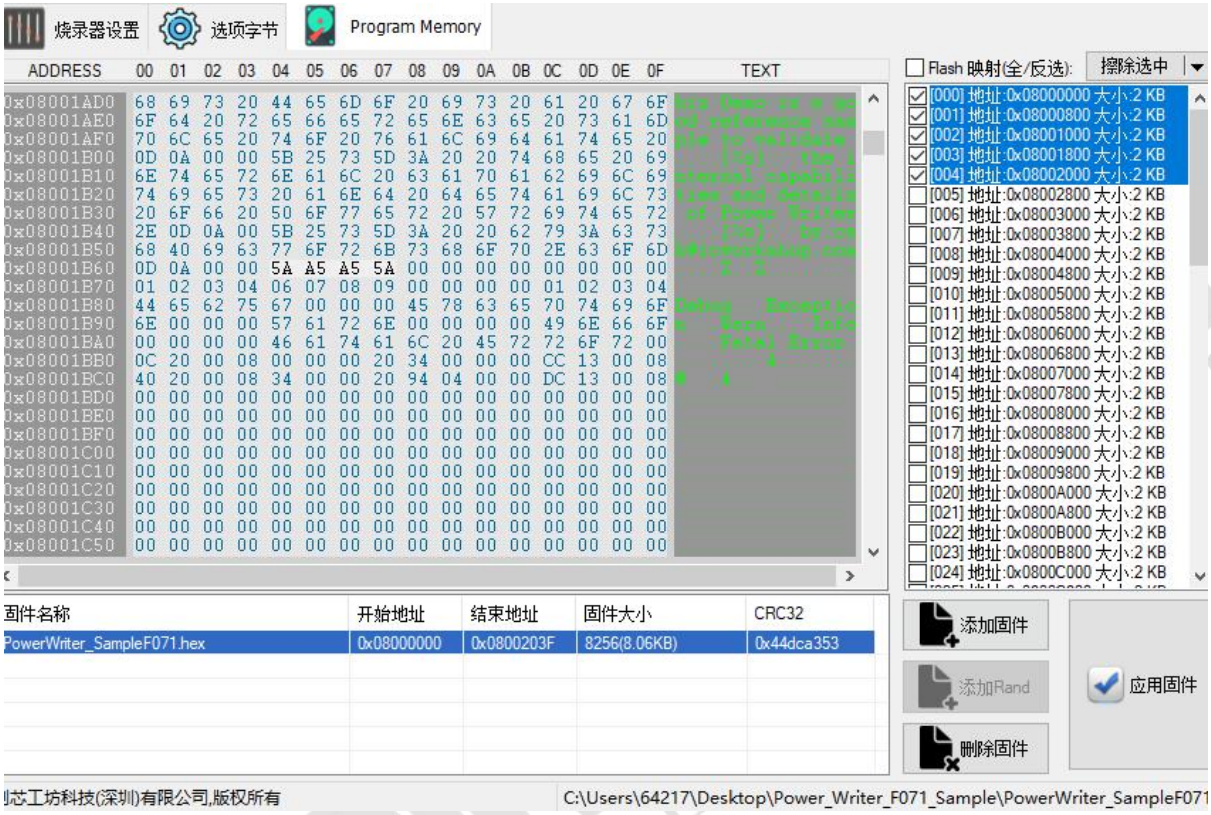


图 4.2.2-3 添加编译出的固件到 Power Writer 的 Program Memory

接下来就是最后一步，我们将项目保存，以便后续的使用，并同步加载到 Power Writer 端，点击菜单的保存并离线加载菜单，如图 4.2.2-4 所示：

图 4.2.2-4 将项目保存到本地并离线加载到 Power Writer

4.2.3 Power Writer 标准离线固件验证

■ 经过上面的两个步骤，我们清晰的熟悉了 Power Writer 固件的开发流程，比如在项目中加入 sn 的技巧，加入 Power Writer 内置 Matrix 算法的流程，并介绍 Power Writer 离线固件的打包流程和加载方法，本节我们将验证我们上次设置固件的信息。加载固件到 Power Writer 之后，我

们按一下 Power Writer 上的离线烧录按钮，启动离线烧录。通过图 4.2.3-1 可以看出，固件的 SN 已经变成 0x00000010 ，Matrix 的验证已经变成了 pass，说明 Power Writer 写入的信息是正确的。其他的 Power Writer 选项验证请自行测试。

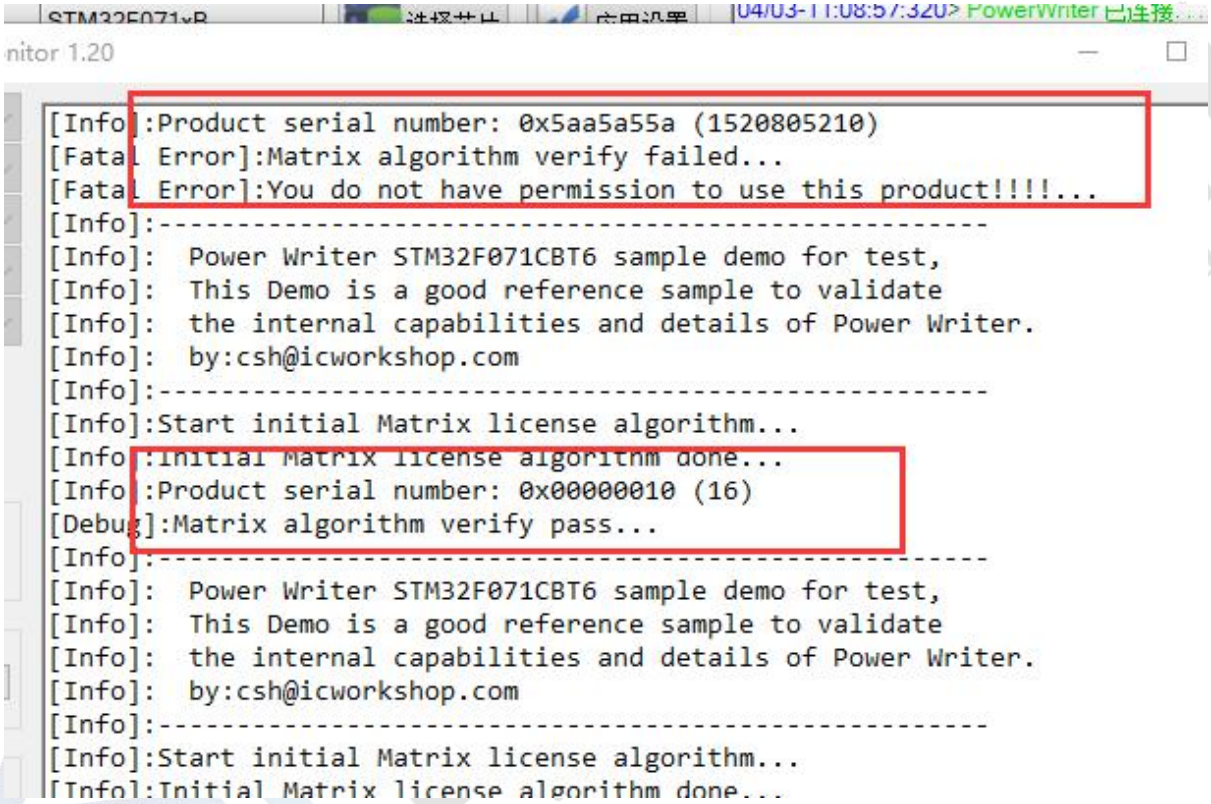


图 4.2.3-1 Power Writer 验证离线固件的功能

4.2.4 Power Writer ICWKEY 授权应用流程

基于前面几节我们验证了 Power Writer 的开发者功能，以及离线量产，内部 Matrix 授权算法等功能，本节演示 Power Writer 配合 ICWKEY 的授权应用场景，ICWKEY 为创芯工坊开发的专用安全授权工具，首先我们将 ICWKEY 的 SDK 加入我们的项目，SDK 的存放路径在 ICWKEY 应用软件的根目录下的 SDK 目录下，如图 4.2.4 -1 所示：

图 4.2.4 -1 ICWKEY 的 SDK 所在路径

我们将 sissdk 整个目录拷贝到我们的项目路径下，如图 4.2.4 -2 所示：

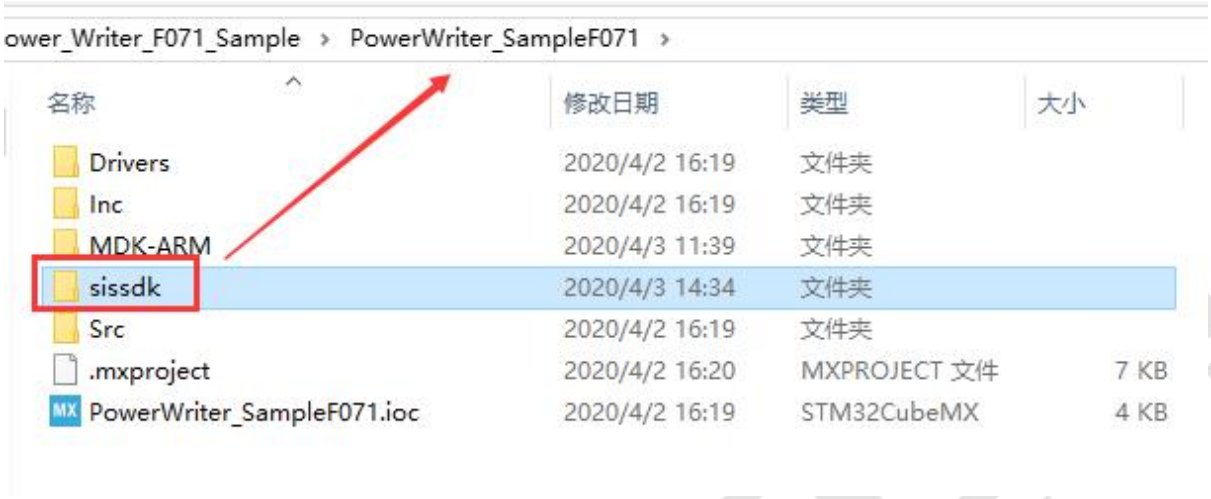


图 4.2.4 -2 拷贝 sissdk 目录到工程项目

在 sissdk/configure 文件夹中有一个截图说明了加入 sissdk 需要添加的需要的文件路径到 keil 中，如图 4.2.4-3 所示：

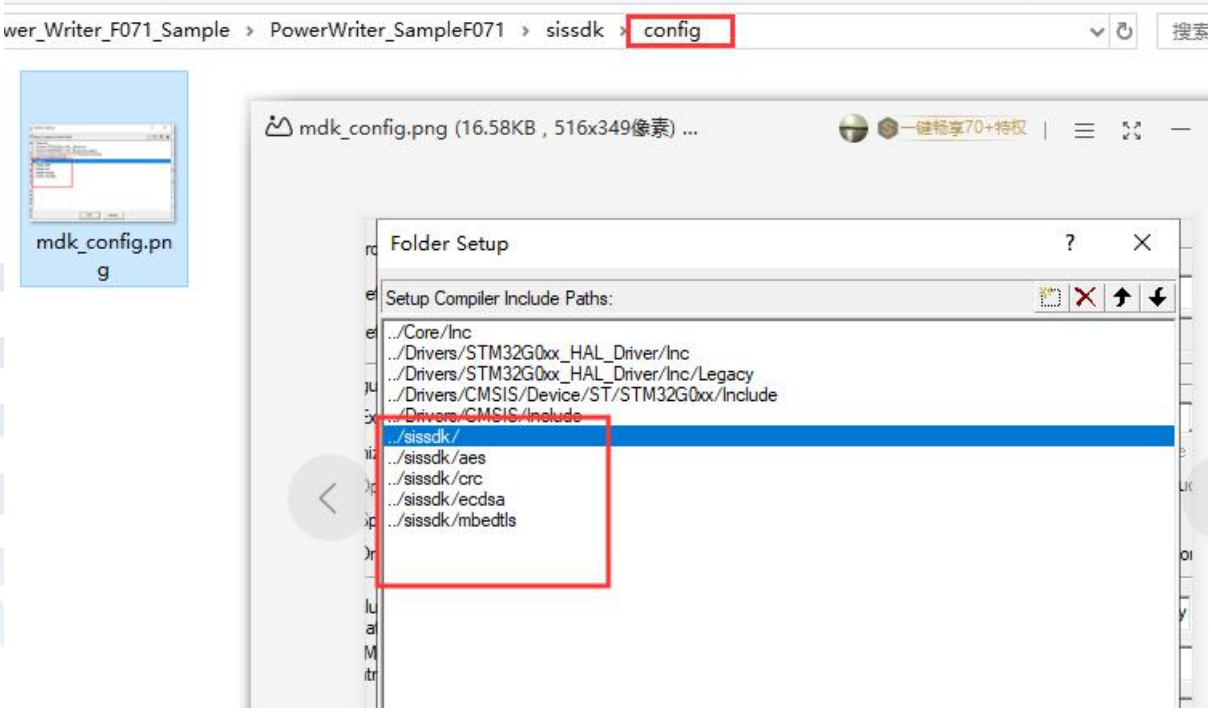


图 4.2.4-3 SDK 需要配置的路径

配置好路径之后，将 sissdk 的 source 添加到项目中，其中 mbedtls 为 arm 开发的轻量级加密库，这是一个跨平台的加密库，并且拥有极高的性能，这里是 power writer 采用此加密库用于项目开发的首选加密库，配置好路径，添加文件到项目中之后，目录结构如图 4.2.4-3 所示：

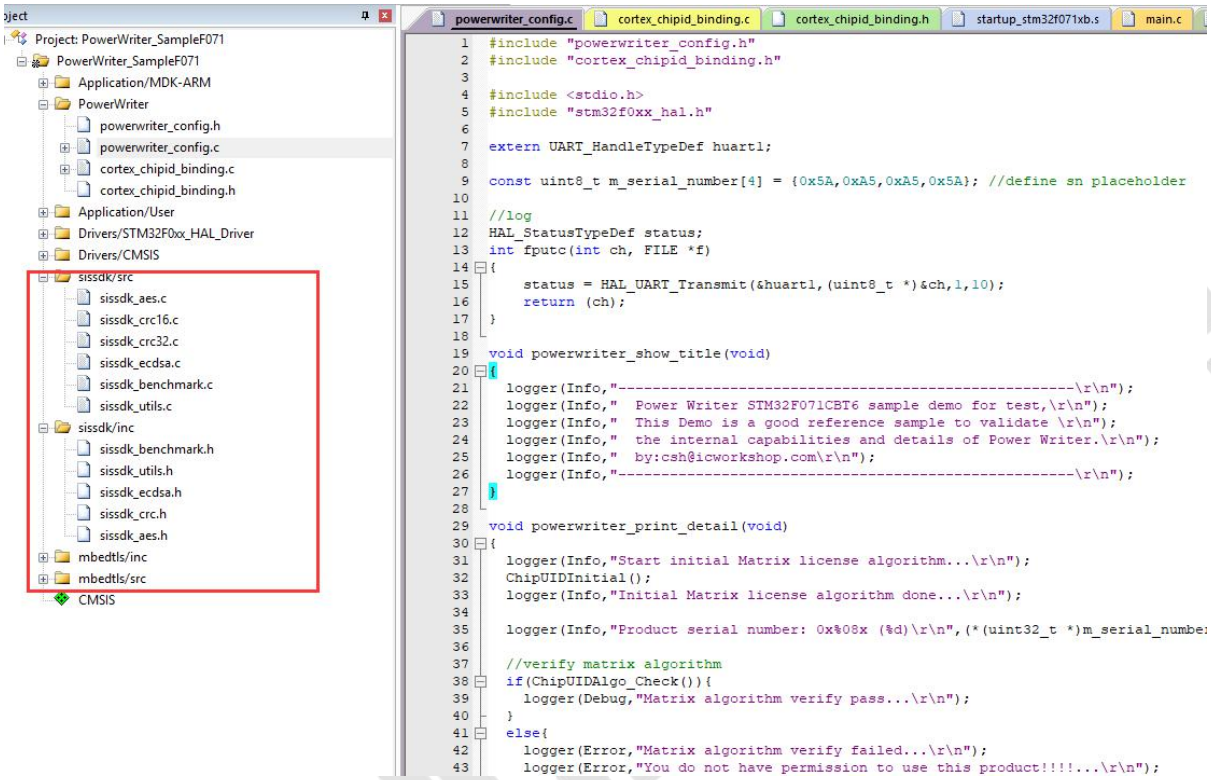


图 4.2.4-3 将 ICWKEY 加入项目中的目录结构

接下来我们通过 Power Writer 开启 ICWKEY 授权，在 UID 加密设置选项卡中选择加密模式为 ICWKEY 模式，然后随机生成通信密码，填写项目名称，如图 4.2.4-4 所示：

图 4.2.4-3 Power Writer ICWKEY 授权方式设置界面

然后我们来到打开 ICWKEY 的应用软件，并用 Type c 的数据线连接到电脑，使用默认的密码和项目名称(如果是重复使用的 ICWKEY 使用真实的项目名称和密码进行连接，如果找不到设备，请安装 ICWKEY 的驱动，参考 ICWKEY 用户参考手册.pdf)，连接成功之后，我们需要将 Power Writer 端的密码和项目名称更新到 UKEY 端，如图 4.2.4-4 所示：

图 4.2.4-4 ICWKEY 同步设置 Power Writer 端设置

然后我们切换选项卡到 UID 算法部分，选择 ECDSA 签名方式，向量矩阵 Matrix 和 Power Writer 内置的授权算法是一样的，我们之前已经测试过了，这里不重复测试，选择 ECDSA 之后，我们点击生成签名证书按钮，将看到证书生成界面，如图 4.2.4-4 所示，然后我们分别执行随机生成-》导出源码-》编译并保存按钮：

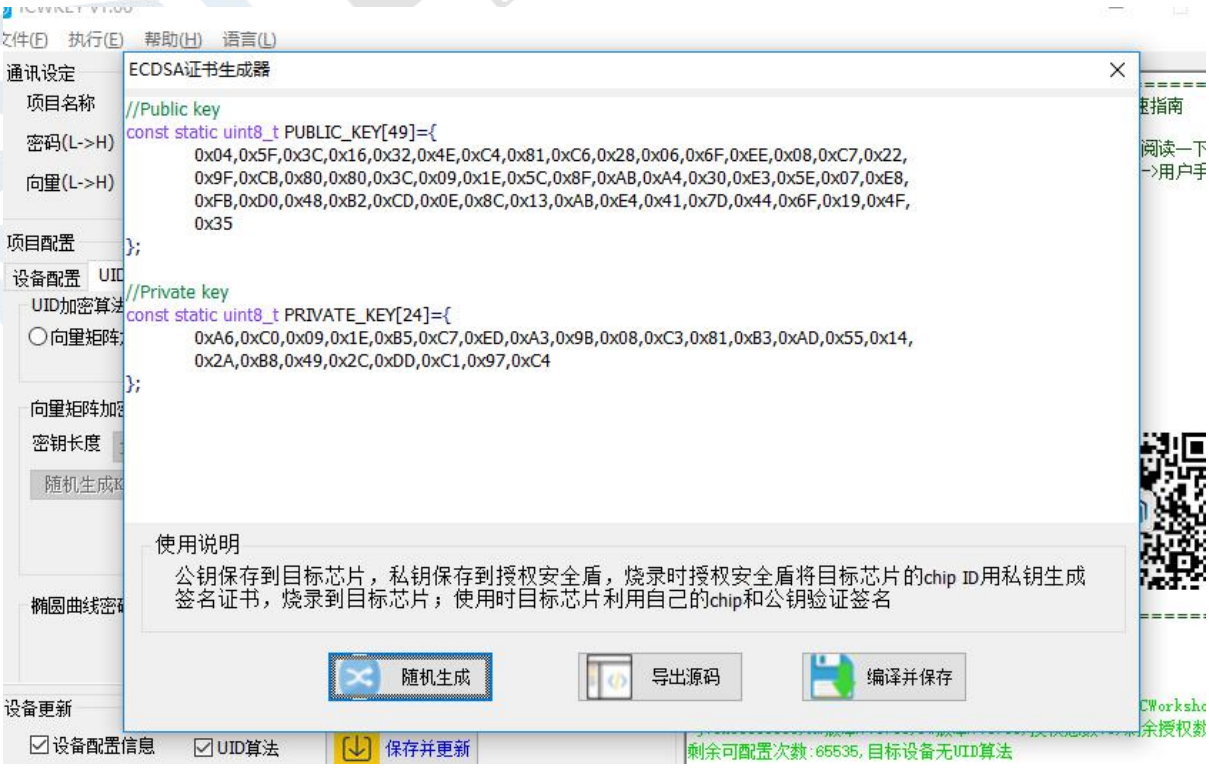


图 4.2.4-4 ICWKEY 配置证书

来到主页面，我们通过保存最终的设置到 ICWKEY 端，通过点击按钮 保存并更新，同步设置时同步会保存项目，需要设置项目密码，如图 4.2.4-5 所示：

图 4.2.4-4 同步所有设置到 ICWKEY

接下来我们需要将 ICWKEY 导出的源码加入到我们的项目工程中，再加入 ECDSA 验证源码时，需要留意 CHIPID 的初始化方式，有些芯片的 UID 是不连续的(STM32L1_Series 和 STM32L0_Series 等)，同时要设置签名的存储地址，Demo 中设置签名地址为 0x08004000,其次是要设置 CHIP ID 的地址 UID_CHIP_ADDR,当一切准备就绪之后，我们加入测试代码，如图 4.2.4-5 所示：

```
//verify ecdsa sign
sis_log(LogError,"Initial ECDSA Digital e-cert algorithm ...\r\n");
ChipUIDInitial_ECDSA();
if(ChipUIDAlgo_Check_ECDSA()){
  sis_log(LogDebug,"ECDSA Digital e-cert verify pass...\r\n");
}
else{
  sis_log(LogError,"ECDSA Digital e-cert verify failed...\r\n");
  sis_log(LogError,"You do not have permission to use this product!!!!...\r\n");
}
}
```

图 4.2.4-5 在项目中加入 ECDSA 数字电子证书签名代码

加完所有代码之后，我们看下 Demo 默认输出的日志，如下图所示

```
[Info]:-------------------------------------------------
[Info]:  Power Writer STM32F071CBT6 sample demo for test,
[Info]:  This Demo is a good reference sample to validate
[Info]:  the internal capabilities and details of Power Writer.
[Info]:  by:csh@icworkshop.com
[Info]:-------------------------------------------------
[Debug]:Crc32 test
[Debug]:Crc32:0xcdb8a549                    CRC32 测试
[32][ok]: Hash = D0B77A82CE689D5E23DD56080C445F601BFE9DFC0423BBD822B81E9742B65DE0
[Debug]:[-s]Signing src hash...
[Debug]:[ok]: ok (signature length = 54)    生成数字电子证书测试
[54][ok]: Signature = 303402181756623E641890BC84E65C12D7711EB6D77F92D8C1CC3D0C0218726F59200FB894DCBAEC992
[Debug]:Sign ok...
[Debug]:---------------Elapsed time : 0 ms ---------------
[32][ok]: Hash = D0B77A82CE689D5E23DD56080C445F601BFE9DFC0423BBD822B81E9742B65DE0
[Debug]:[-v]Preparing verification context...
[Debug]:[ok]: verification passed...
[Debug]:verify passed...                    数字电子证书校验测试
[Debug]:---------------Elapsed time : 0 ms ---------------
[Info]:Start initial Matrix license algorithm...
[Info]:Initial Matrix license algorithm done
[Info]:Product serial number: 0x5aa5a55a (1520805210)   序列号为初始序列号0x5AA5A55A
[Fatal Error]:Matrix algorithm verify failed...
[Fatal Error]:You do not have permission to use this product!!!!...
[Fatal Error]:Initial ECDSA Digital e-cert algorithm ...   Matirx 测试失败不通过 没有进行签名
[32][ok]: Hash = 067FD4C509DDEBE191EC2B62973C1E816A071D72C3B25A925482F2297AFC653C
[Debug]:[-v]Preparing verification context...
[Fatal Error]:[failed]:mbedtls_ecdsa_genkey returned -20450
[Fatal Error]:ECDSA Digital e-cert verify failed...   ECDSA 电子签名失败，没有进行证书写入
[Fatal Error]:You do not have permission to use this product!!!!...
```

图 4.2.4-5 Demo 直接写入芯片是没有序列号、Matrix 证书、ECDSA 证书的

所有代码加完之后，我们打开 PowerWriter_SampleF071.map 文件，再次核对我们加入信息的 Flash 地址：

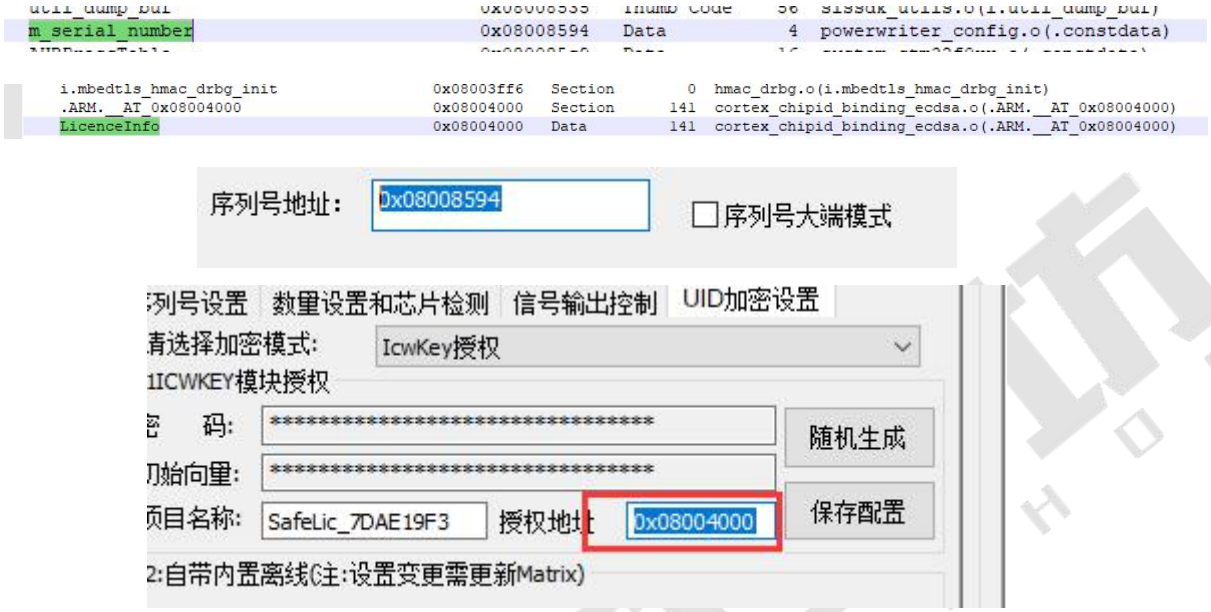


图 4.2.4-5 核对最终项目编译的变量在 Flash 中的地址并更新到 PowerWriter

序列号 SN 地址为：0x08008594

ECDSA 电子证书地址为：0x08004000

完成设置之后保存，将 Power Writer 项目加载到 PowerWriter，并使用 ICWKEY 和 PowerWriter 进行连接，当 ICWKEY 和 PowerWriter 连接成功之后将会听到“滴滴两声”，烧录完毕之后，我们再查看一下 串口输出日志内容，如图 4.2.4-6 所示：

```
[Info]:-----------------------------------------------------
[Info]:  Power Writer STM32F071CBT6 sample demo for test,
[Info]:  This Demo is a good reference sample to validate
[Info]:  the internal capabilities and details of Power Writer.
[Info]:  by:csh@icworkshop.com
[Info]:-----------------------------------------------------
[Debug]:Crc32 test
[Debug]:Crc32:0xcdb8a549
[32][ok]: Hash = D0B77A82CE689D5E23DD56080C445F601BFE9DFC0423BBD822B81E9742B65DE0
[Debug]:[-s]Signing src hash...
[Debug]:[ok]: ok (signature length = 54)
[54][ok]: Signature = 3034021817566236641890BC84E65C12D7711EB6D77F92D8C1CC3D0C0218726F59200FB894DCBAEC992
[Debug]:Sign ok...
[Debug]:---------------Elapsed time : 0 ms ---------------
[32][ok]: Hash = D0B77A82CE689D5E23DD56080C445F601BFE9DFC0423BBD822B81E9742B65DE0
[Debug]:[-v]Preparing verification context...
[Debug]:[ok]: verification passed...
[Debug]:verify passed...
[Debug]:---------------Elapsed time : 0 ms ---------------
[Info]:Start initial Matrix license algorithm...
[Info]:Initial Matrix license algorithm done...
[Info]:Product serial number: 0x00000000 (0)        序列号已更新
[Fatal Error]:Matrix algorithm verify failed...
[Fatal Error]:You do not have permission to use this product!!!!...
[Fatal Error]:Initial ECDSA Digital e-cert algorithm ...
[32][ok]: Hash = 067FD4C509DDEBE191EC2B62973C1E816A071D72C3B25A925482F2297AFC653C
[Debug]:[-v]Preparing verification context...
[Debug]:[ok]: verification passed...                电子证书ECDSA 验证通过
[Debug]:ECDSA Digital e-cert verify pass...
```

图 4.2.4-6 数字电子证书 ECDSA 验证

通过以上的演示，我们不仅知道 Power Writer 灵活的烧录功能，内部随机生成 Matrix 证书的功能，更可以通过专业的 ICWKEY 生成 ECDSA 数字电子签名来保护我们的程序，由于 ECDSA 私钥证书在 ICWKEY 内部，公钥在项目中，加上代码做了隐藏地址的操作，基本可以杜绝程序被拷贝的

可能，实现专业级的芯片防护，ICWKEY 开放整个 SDK，用户可以自行根据源码编译，不同的项目可以使用不同的编码格式，简单地集成方法和完善的文档和技术支持。

附加信息：

- ICWKEY 的详细开发指南 请参考《ICWKEY 用户参考手册.pdf》
- Power_Writer_F071_Sample 完整的 Demo code 可以到 创芯工坊官方下载

4.3Power Writer 创芯工坊应用

这一章我们介绍在创芯工坊中应用 Power Writer 进行生产控制，Power Writer 的远程量产控制能力，主要可以分为四类应用：

- **批量离线**：大批量标准模式
- **在线 ECDSA 数字电子签名**:创芯工坊提供授权服务器,SDK 基于 sissdk 开发
- **在线自定义授权算法**：创芯工坊提供授权服务器，内置 Matrix 模板
- **用户自建授权服务器**：创芯工坊提供授权服务器搭建模板

下面我们分别大致介绍下各种模式下的流程。

4.3.1 Power Writer 批量离线模式

通过 3.4 节的内容我们清楚了创芯工坊的注册以及使用流程，这里我们不再描述注册以及客户端安装的问题，我们直接上传我们在 4.2 节 准备的测试项目固件上传到创芯工坊后台，参数设置如图 4.3.1-1 所示。

图 4.3.1-1 上传发布设置

固件上传参数页设置如图 4.3.1-2 所示，我们的测试项目是基于 **STM32F071CB** 芯片的，所以在这里，我们选择 ST->STM32F0 系列-> STM32F07xB-> **PW200@PowerWriter(标准版本的内部代号)**，接下来我们上传通过 Power Writer 打包的项目文件，填写保存 PowerWriter 项目时输入的密码，选择

编程模式为离线模式，检查无误之后，点击页面底部的 确认发布 按钮，将订单发布到创芯工坊后台服务器。

图 4.3.1-2 Power Writer 离线远程离线生产模式参数示范

项目发布后，我们登录创芯工坊的客户端，并将 Power Writer 和 PC 保持连接(如果连接不上，先安装驱动，安装方法见 使用前的准备)，登录到创芯工坊，我们将看到我们刚才发布的**生产订单,** 如图 4.3.1-3 所示：

图 4.3.1-3 通过创芯工坊客户端登录账号查看生成订单

接下来我们将此订单加载到 Power Writer 中，我们选择刚才的订单，点击创芯工坊界面下面的 下载固件 按钮，将弹出固件量产设置对话框，如图 4.3.1-4 所示：

图 4.3.1-4　Power Writer 的烧录参数配置

- **编程模式：** 此处将提示我们订单的类型为离线模式
- **烧录次数：** 表示此处下载将消耗多少次固件量产次数，从订单中扣除，用完不能再用
- **设置序列号：** Power Writer 项目中的序列号是固定的，在这里可以分段设置序列号的起始信息（项目中开启 SN 才生效）

我这里设置了 100 次离线烧录次数，设置序列号的起始为 0x00000040,然后点击确定按钮，等待片刻，创芯工坊将此订单加载到 Power Writer 中了，显示结果如图 4.3.1-5 所示：

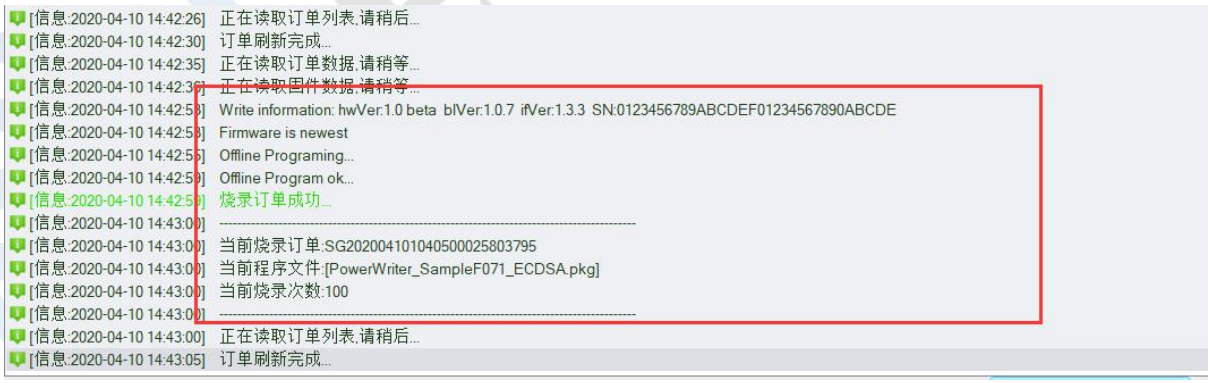


图 4.3.1-5　创芯工坊将 Power Writer 项目以离线方式加载到 Power Writer

由于我们选择的是离线烧录方式，我们需要手动按一下 Power Writer 上的按钮(生产时是通过 CTRL 信号控制)，烧录完成之后，我们连接串口到芯片上，可以看到输出日志的提示 SN 和 ECDSA 证书验证都已通过，如图 4.3.1-6 所示

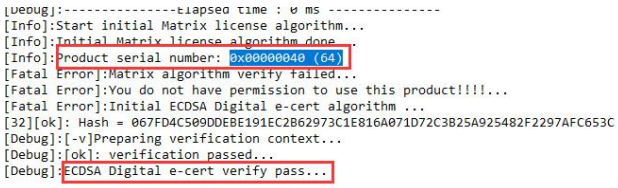


图 4.3.1-6 创芯工坊 Power Writer 离线方式验证结果

4.3.2 Power Writer 在线授权(ECDSA 数字电子证书)

接下来我们测试下 ECDSA 远程服务器的在线固件签名认证模式，流程和离线模式整体是一致的，区别在于需要再授权服务器中添加授权算法，我们按照步骤演示一下流程。

- **第一步：我们在授权中心添加算法，并且创建授权项目：**
  https://www.icworkshop.com/api/user/serviceChipAuth (快速进入授权服务器)，或者点击开发者中心的 **授权中心连接，**我们在算法列表中添加一个 ECDSA 的标准算法，如图 4.3.2-1 所示,点击随机生成证书按钮，并且将公钥下载到本地。

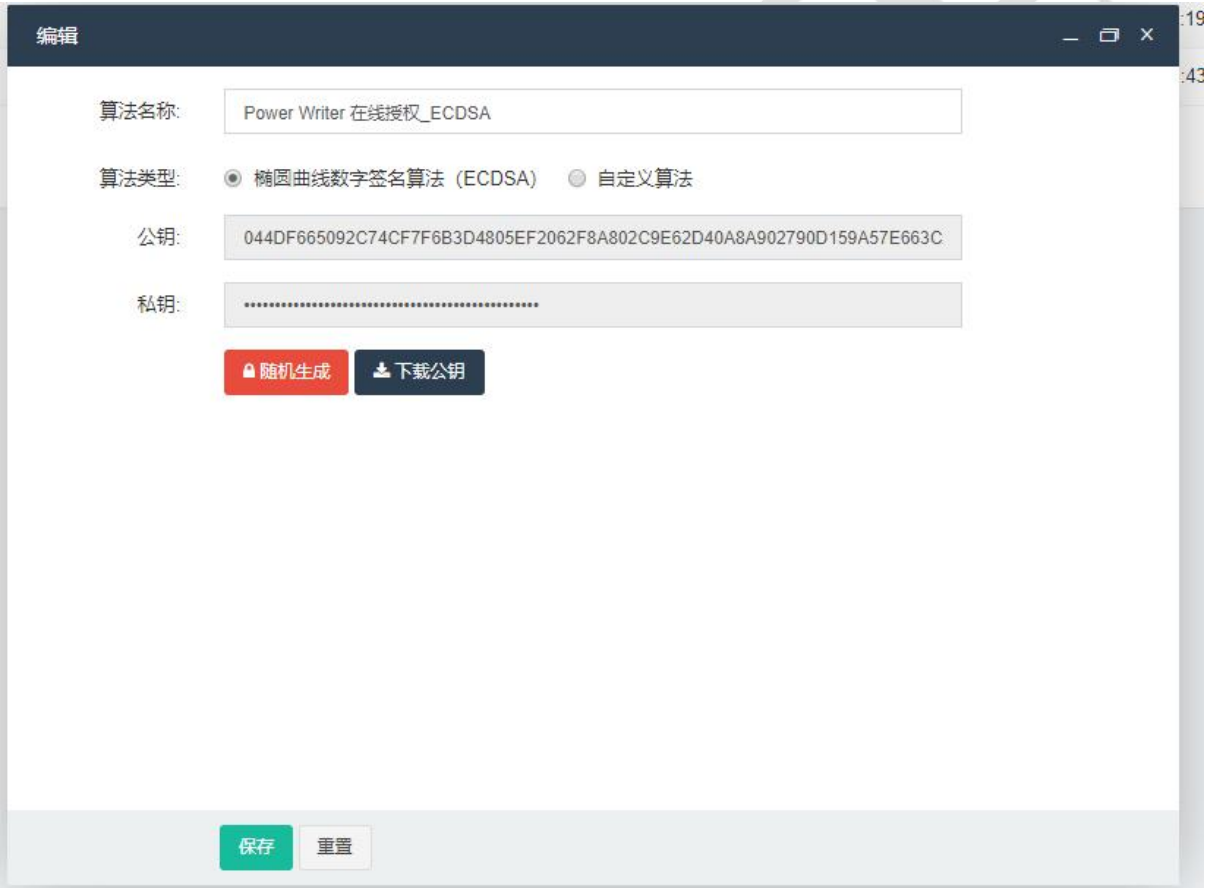


图 4.3.2-1 授权服务器添加 ECDSA 算法

创建好算法之后，我们再次创建一个授权项目，如图 4.3.2-1_1 所示，完成以上两部，就准备妥当了 ECDSA 在线数字证书签名的设置，接下来我们将公钥参考 ICWKEY 的开发方式开发我们的项目。

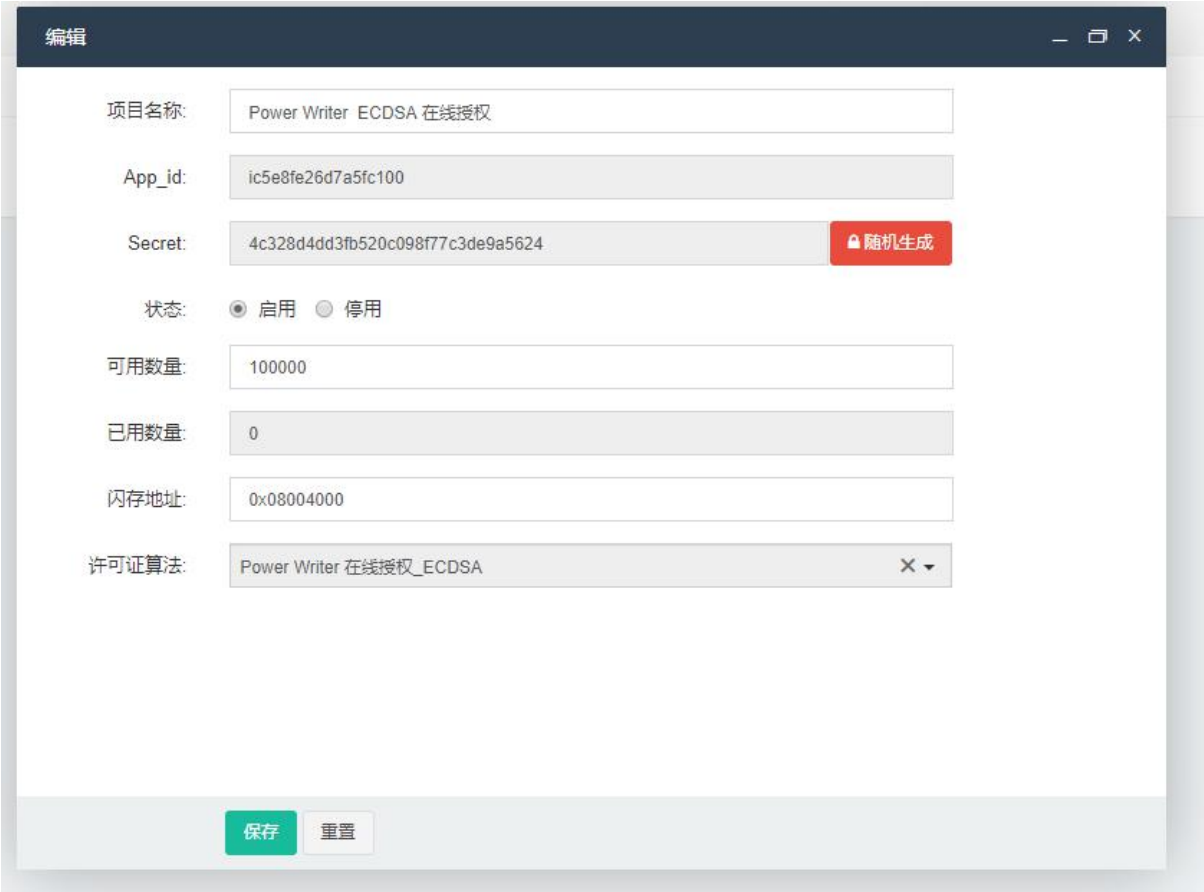


图 4.3.2-1_1 创建 ECDSA 授权项目

- **第二步：我们将公钥导入到项目源码中编译：**为了做芯片授权验证，我们需要价格授权服务器中的公钥导入到 MDK 项目中编译，并重新打包成在手 ECDSA 签名的项目，如图 4.3.2-2 所示，在线 ECDSA 授权的硬件端 SDK 同 ICWKEY 的 SDK，只不过服务器在线授权是远程服务器授权方式，ICWKEY 是本地方式，两者各有优势。

  - **ICWKEY 方式：**速度更快，但是如果需要手机日志，生成报表则不那么方便，并且无法在线进行权限控制，针对大批量生产模式
  - **ECDSA 授权服务器模式：**速度较慢，每次只能烧一颗，但是可以后台控制权限，并且可以完整生成报表，针对版权保护较为严格的高技术含量项目

```
/* Includes ------------------------------------------------------------------*/
#include "cortex_chipid_binding_ecdsa.h"

//Public key for icwkey demo
/*
const static uint8_t PUBLIC_KEY[49]={
  0x04,0xDE,0x47,0xBA,0x28,0x94,0x60,0x7A,0x5D,0x1F,0x8E,0x4B,0xC2,0x72,0xF2,0xA3,
  0x31,0xBF,0x97,0x2E,0x14,0x3B,0xA9,0x47,0x76,0xBD,0xE0,0xA5,0x7D,0x4F,0x98,0x53,
  0xD7,0x86,0x09,0x99,0x40,0xD9,0xBF,0x38,0x3F,0xBC,0x4C,0x09,0xE0,0xD9,0x2F,0x39,
  0x8A
};*/
                                                    授权服务器生成的公钥
//Public key for icworkshop online license server
const static uint8_t PUBLIC_KEY[49]={
  0x04,0x4D,0xF6,0x65,0x09,0x2C,0x74,0xCF,0x7F,0x6B,0x3D,0x48,0x05,0xEF,0x20,0x62,
  0xF8,0xA8,0x02,0xC9,0xE6,0x2D,0x40,0xA8,0xA9,0x02,0x79,0x0D,0x15,0x9A,0x57,0xE6,
  0x63,0xCA,0x43,0x49,0xD4,0xBF,0x59,0xEF,0xC9,0x2E,0x46,0xEE,0x40,0x3D,0xFD,0xE6,
  0x23
};
```

图 4.3.2-2 创芯工坊授权服务器生成的公钥

- **第三步：重新打包 Power Writer 项目：**我们需要将项目的授权模式选择为在线方式，并且在服务端也要设置为在线模式。如图 4.3.2-3 所示：

图 4.3.2-3 PowerWriter 打包设置为在线授权

注：在线授权,是可选的，服务端依然可以按照离线的方式发布项目，这样等同于关闭了 Power Writer 的内部 Matrix 、ICWKEY 的 Matrix、ECDSA 以及所有服务端的授权方式。也就是不使用任何一种授权方式来保护我们的项目。

- **第四步：发布项目是服务端选择在线授权：**由于我们采用在线授权来授权我们项目，所以我们需要再服务端设置为在线模式，如图 4.3.2-4 所示：

图 4.3.2-4 在线 ECDSA 签名设置参考

- **第五步：创芯工坊在线 Power Writer 烧录：**当我们按照设置将项目发布之后，我们就准备好了测试所有的资料，接下来登录创芯工坊，设置界面和离线界面有所区别，序列号暂时只能一颗一颗的设置(SN 开启的情况下)，并且每次只能烧录一颗芯片（在线授权，后续会增加批量在线授权，待用户反馈更新），设置界面如图 4.3.2-4 所示：

图 4.3.2-4 在线 ECDSA 烧录设置

- **第六步：验证 ECDSA 签名结果，**我们通过上面的设置将项目远程下载到目标芯片，并从授权服务器获取 ECDSA 签名，下载完成后会自动复位(离线固件有复位选项)，我们从串口中可以看到输出的日志内容。如图 4.3.2-5 所示：

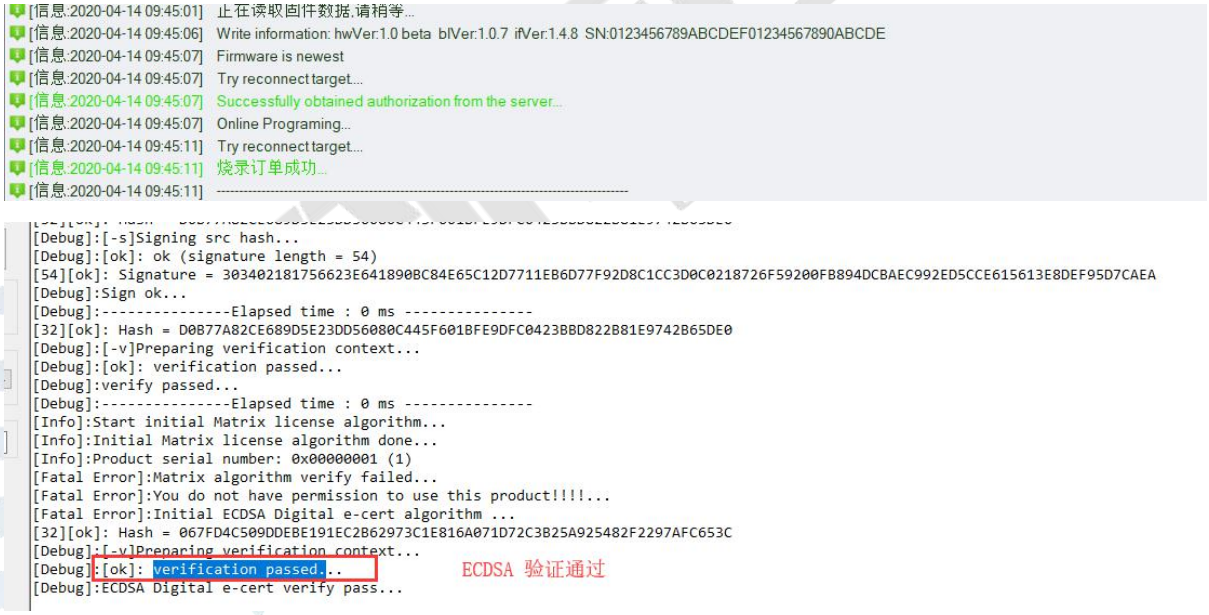


图 4.3.2-5 在线 ECDSA 验证结果

通过以上几步，我们可以很容易的在我们的项目中集成创芯工坊的在线授权方案，并实现远程在线烧录，极大的方便开发者。

注：测试项目源码，打包好的 pkg 项目文件，或者是测试订单均可以在创新工坊下载和索取，方便用户使用测试。

4.3.3 Power Writer 在线授权(自定义授权算法)

接下来我们测试下 **用户自定义授权算法** 远程服务器的在线固件签名认证模式，**流程和在线**

ECDSA 证书签名 是一致的，区别在于 ECDSA 是标准的公钥体系，而自定义允许用户自行实现授权算法。

注:自定义算法需要用户有一定的 php 语言基础，用于编写授权算法函数

- **第一步：我们在授权中心添加算法，并且创建授权项目：**
  https://www.icworkshop.com/api/user/serviceChipAuth (快速进入授权服务器)，或者点击开发者中心的 **授权中心连接**，我们在算法列表中添加一个自定义算法，如图 4.3.3-1 所示，为了方便起见，这边编写的授权算法就是内部 Matrix 矩阵算法的服务端实现：

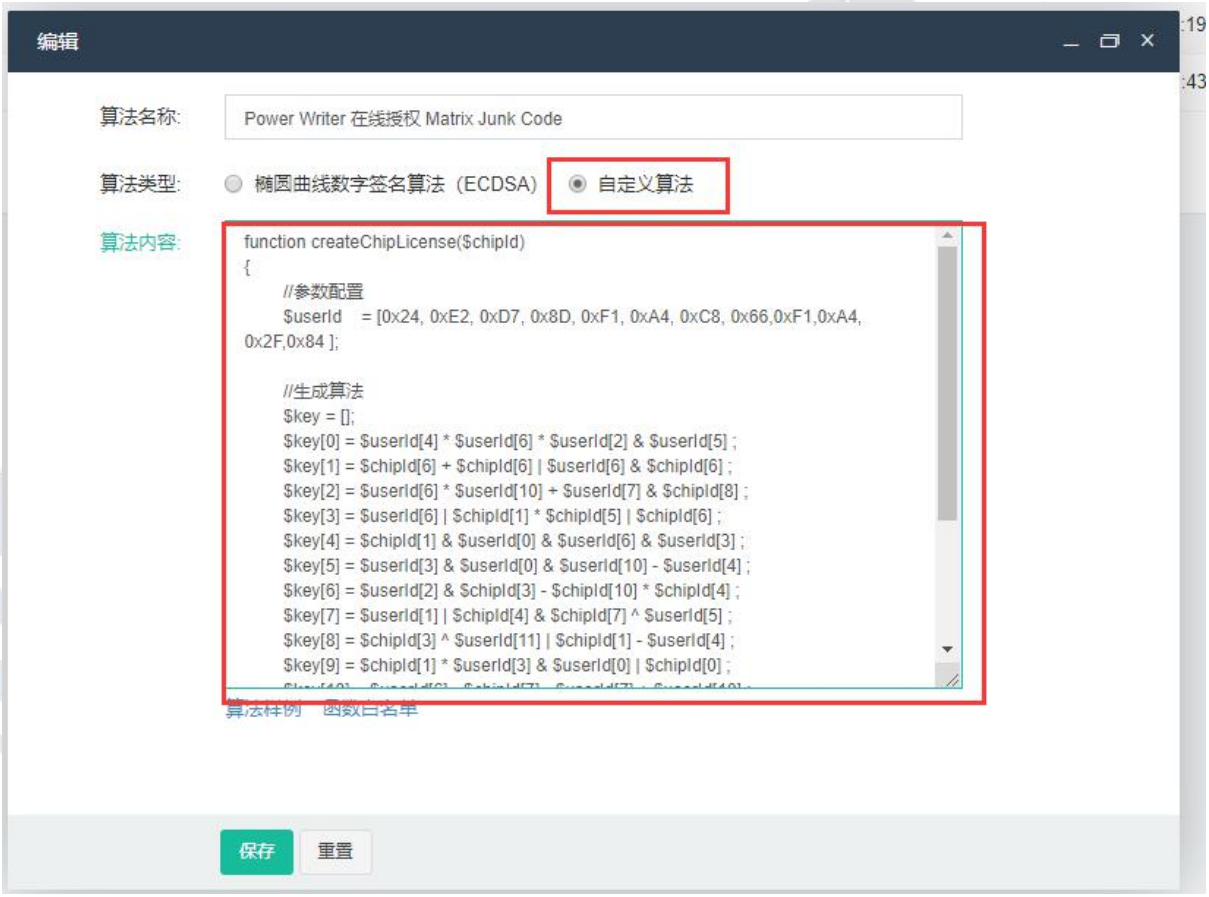


图 4.3.3-1 授权服务器自定义算法的 PHP 实现

```
88
89  #else
90  //The following code may warn in KEIL(MDK), ignore it
91  static void ChipUIDAlgo(char pUserID[], char pChipID[], char pKey[])
92  {
93    pKey[0] = pUserID[4] * pUserID[6] * pUserID[2] & pUserID[5] ;
94    pKey[1] = pChipID[6] + pChipID[6] | pUserID[6] & pChipID[6] ;
95    pKey[2] = pUserID[6] * pUserID[10] + pUserID[7] & pChipID[8] ;
96    pKey[3] = pUserID[6] | pChipID[1] * pChipID[5] | pChipID[6] ;
97    pKey[4] = pChipID[1] & pUserID[0] & pUserID[6] & pUserID[3] ;
98    pKey[5] = pUserID[3] & pUserID[0] & pUserID[10] - pUserID[4] ;
99    pKey[6] = pUserID[2] & pChipID[3] - pChipID[10] * pChipID[4] ;
100   pKey[7] = pUserID[1] | pChipID[4] & pChipID[7] ^ pUserID[5] ;
101   pKey[8] = pChipID[3] ^ pUserID[11] | pChipID[1] - pUserID[4] ;
102   pKey[9] = pChipID[1] * pUserID[3] & pUserID[0] | pChipID[0] ;
103   pKey[10] = pUserID[6] - pChipID[7] - pUserID[7] + pUserID[10] ;
104   pKey[11] = pUserID[4] & pChipID[3] | pUserID[6] + pChipID[5] ;
105
106 }
107
108 #endif
109
```

图 4.3.3-2 硬件端采用和服务端同样的验证算法的 C 语言实现

创建好算法之后，我们再次创建一个授权项目，如图 4.3.3-1_1 所示，完成以上两部，就准备妥当了自定义算法签名的设置，

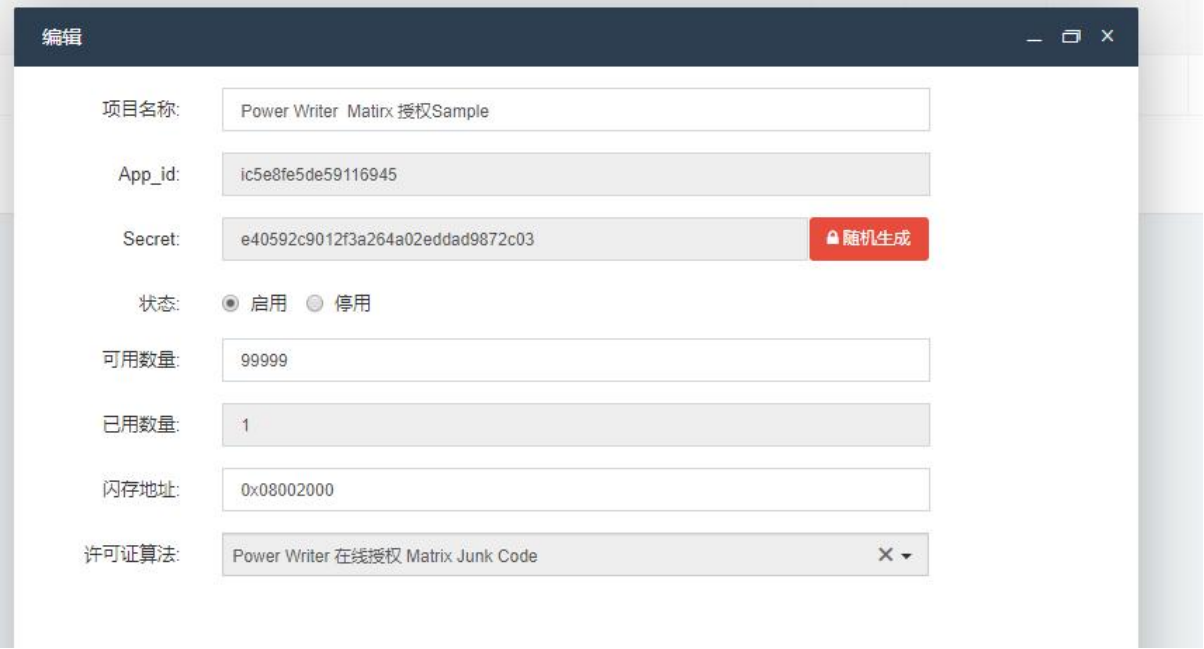


图 4.3.3-1_1 创建自定义授权项目

- **第二步：重新打包 Power Writer 项目：**我们需要将项目的授权模式选择为在线方式，并且在服务端也要设置为在线模式。如图 4.3.2-3 所示：

图 4.3.3-3 PowerWriter 打包设置为在线授权

- **第三步：发布项目是服务端选择在线授权：**由于我们采用在线授权来授权我们项目，所以我们需要再服务端设置为在线模式，如图 4.3.3-4 所示：

图 4.3.3-4 在线自定义算法签名设置参考

- **第四步：创芯工坊在线 Power Writer 烧录：**当我们按照设置将项目发布之后，我们就准备好了测试所有的资料，接下来登录创芯工坊，设置界面和离线界面有所区别，序列号暂时只能一颗一颗的设置(SN 开启的情况下)，并且每次只能烧录一颗芯片，设置界面如图 4.3.3-5 所示：

图 4.3.2-5 设置界面跟 ECDSA 是一样的

- **第五步：验证自定义算法签名结果，**我们通过上面的设置将项目远程下载到目标芯片，并从授权服务器获取自定义算法的签名**(demo 用的 Matrix 方式)**，下载完成后会自动复位(离线固件有复位选项)，我们从串口中可以看到输出的日志内容。如图 4.3.2-5 所示：

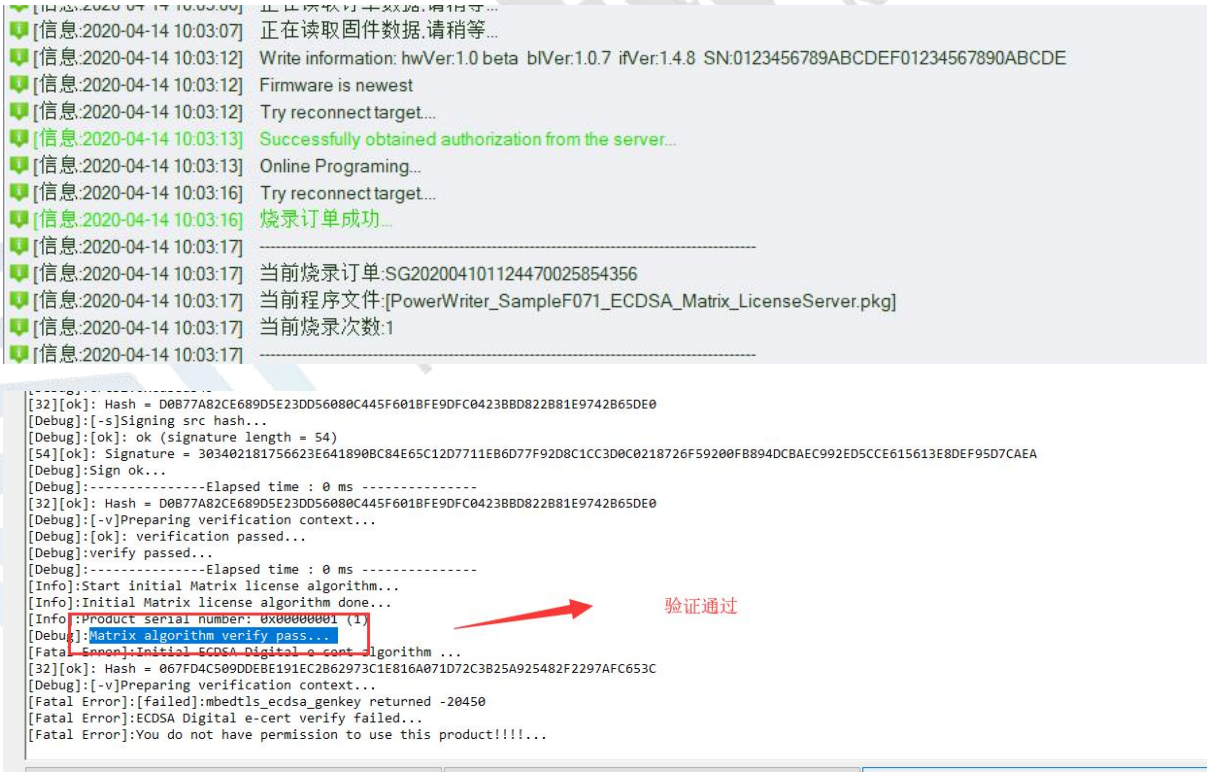


图 4.3.2-5 在线 ECDSA 验证结果

在线授权方式的两种：自定义授权算法和在线 ECDSA 数字证书电子签名 两者的开发流程是类似的，区别在于 ECDSA 授权采用的是 ICWKEY 的 SDK 开发，其实是一个**线上版本的 ICWKEY,**自定义授权算法由用户自行实现授权代码，实现与创芯工坊平台完全分离，创芯工坊只是作为一个中间平台角色来量产用户的订单，开发者可完全掌控核心技术，包括授权机制。

4.3.4 Power Writer 用户自建授权服务器方法

创芯工坊官方提供了内建服务器的模板，如果用户想自建授权服务器，Power Writer 也同样支持，自建授权服务器首先访问自建授权服务器帮助文档，按照流程将授权服务器搭建起来，由于自建服务器的话，需要做二次开发，创芯工坊提供了授权服务器的快速开发模板，https://www.icworkshop.com/article/helpCenter/94/36，如图 4.3.4-1 所示

图 4.3.4-1 自建授权服务器的流程

服务器二次开发完成之后，在上传 Power Writer 固件到创芯工坊时，编程模式选择在线授权(自有授权)模式，并填好授权 API 的 HTTPS 地址和 API 请求认证密钥，如图 4.3.4-2 所示：

图 4.3.4-2 自建授权服务器的参数选择

注:自建授权服务器针对有服务端开发能力的客户，创芯工坊提供快速搭建模板,用户可以开发出功能完善的授权服务器，并使用创芯工坊进行产品量产。

5 ： Power Writer FAQ 常见问题以及解决方法

本章节将累计更新用户在使用 Power Writer 可能问到地比较频繁的问题，创芯工坊负责整理搜集并同步更新到文档，以方便用户查阅。

整理中… …

6 ： 附录

以下章节为开发者或者用户可能需要用到的常用表格汇总，单独作为一章节，方便用户查阅，以及参考，随着 Power Writer 对适配的不断进行，表会同步更新，具体请留意创芯工坊官方的升级日志。

6.1Power Writer Flash 起始地址对齐表

表 7.1 为常用芯片的基地址对齐信息汇总，PowerWriter 大部分支持单字节对齐，也即无须对齐，Power Writer 对于没有对齐到下表的用户固件，回读数据并对齐写入，此功能绝大部分时候都是可行的，但部分芯片不支持多次写入，比如 STM32H7，为了保证用户能正常烧写芯片，按表 7-1 对齐固件依然是最佳的选择方案，此表仅供查询参考。

芯片厂商	芯片系列	FLASH 基地址	支持对齐方式
ST	STM32F0xx	0x08000000	相对于 FLASH 基地址 2 字节对齐
	STM32F1xx	0x08000000	相对于 FLASH 基地址 2 字节对齐
	STM32F2xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	STM32F3xx	0x08000000	相对于 FLASH 基地址 2 字节对齐
	STM32F4xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	STM32F7xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	STM32L0xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	STM32L1xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	STM32L4xx	0x08000000	相对于 FLASH 基地址 8 字节对齐
	STM32L5xx	0x08000000	相对于 FLASH 基地址 8 字节对齐
	STM32G0xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	STM32G4xx	0x08000000	相对于 FLASH 基地址 8 字节对齐
	STM32WBxx	0x08000000	相对于 FLASH 基地址 8 字节对齐
	STM32H7xx	0x08000000	相对于 FLASH 基地址 32 字节对齐
GD	GD32F10x	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32F1x0	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32F20x	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32F30x	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32F3x0	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32F4xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32E230	0x08000000	相对于 FLASH 基地址 4 字节对齐
	GD32E103	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32F00x	0x08000000	相对于 FLASH 基地址 4 字节对齐

MM32	MM32F0xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32F1xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32L0xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32L3xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32SPINxx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32W0xx	0x08000000	相对于 FLASH 基地址 4 字节对齐
	MM32W3xx	0x08000000	相对于 FLASH 基地址 4 字节对齐

表 7.1 芯片 Flash 对齐信息不完全列表参考 - 包含第三方收集

6.2Power Writer UID 地址对照表格

表 7.2 为常用芯片的 UID 地址信息汇总，Power Writer 支持自动导出源码，用户无须关心 UID 地址的定义，此表仅供查询参考

芯片类型	芯片系列	芯片型号	UID 起始地址
STM32	STM32F0XX	-	0x1FFFF7AC
	STM32F1XX	-	0x1FFFF7E8
	STM32F2XX	-	0x1FFF7A10
	STM32F3XX	-	0x1FFFF7AC
	STM32F4XX	-	0x1FFF7A10
	STM32F7XX	-	0x1FF0F420 0x1FF07A10(F72xx,F73xx)
	STM32G0XX	-	0x1FFF7590
	STM32G4XX	-	0x1FFF7590
	STM32H7XX		0x1FF1E800
	STM32L0XX	-	0x1FF80050 Offset 0x00：UID[0:31] Offset 0x04：UID[32:63] Offset 0x14：UID[64:95]
	STM32L1XX	-	0x1FF80050 (cat.1/2) 0x1FF800D0(cat.3/4/5/6) Offset 0x00：UID[0:31] Offset 0x04：UID[32:63] Offset 0x14：UID[64:95]

	STM32L4XX	-	0x1FFF7590
	STM32L5XX		0x0BFA0590
	STM32WB		0x1FFF7590
GD32	GD32F10X	-	0x1FFFF7E8
	GD32F1X0	-	0x1FFFF7AC
	GD32F20X	-	0x1FFFF7E8
	GD32F30X	-	0x1FFFF7E8
	GD32F3X0	-	0x1FFFF7AC
	GD32F4XX	-	0x1FFF7A10
	GD32E103	-	0x1FFFF7E8
	GD32E230	-	0x1FFF F7AC
STM8	STM8S	STM8S003xx	0x00004865
		STM8S103xx	0x00004865
		STM8S105xx	0x000048CD
		STM8S207xx	0x000048CD
		STM8S208xx	0x000048CD
		STM8S903xx	0x00004865
	STM8L	STM8L001J3	0x00004925
		STM8L101xx	0x00004925
		STM8L15xxx	0x00004926
		STM8L16xxx	0x00004926
	STM8AL	STM8AL31xx/3Lxx	0x00004926
	STM8TL	STM8TL52x4/53x4	0x00004925
MM32	M32F00XX	-	0x1FFFF7E8
	M32F0XX	-	0x1FFFF7E8
	M32L0XX	-	0x1FFFF7E8
	M32L3XX	-	0x1FFFF7E8
	M32W0XX	-	0x1FFFF7E8
	M32W3XX	-	0x1FFFF7E8
	M32SPINXXX	-	0x1FFFF7E8
	M32F10XX	-	0x1FFFF7E8
NORDIC	NRF51XX	-	0x10000060
	NRF52XX	-	0x10000060

表 7.2　UID 起始地址不完全列表参考

6.3Power Writer Flash Bank 信息汇总

分类	型号	切换方式
STM32	STM32F101xF	有双 BANK,无需切换
	STM32F101xG	
	STM32F103xF	
	STM32F103xG	
	STM32F427xI	2M 的只有双 Bank,无需切换
	STM32F429xI	
	STM32F437xI	
	STM32F439xI	
	STM32F469xI	
	STM32F479xI	
	STM32F479xG	1M 有单双 bank 可以通过 Option Byte 切换
	STM32F469xG	
	STM32F439xG	
	STM32F437xG	
	STM32F429xG	
	STM32F427xG	
	STM32F76xxI	2M 有单双 bank 可以通过 Option Byte 切换
	STM32F77xxI	
	STM32F76xxG	1M 有单双 bank 可以通过 Option Byte 切换
	STM32F77xxG	
	STM32F742	有双 Bank,但是只有交换功能.没有单双切换功能
	STM32F743	
	STM32F745	
	STM32F747	
	STM32F750	
	STM32F753	
	STM32F755	
	STM32F757	
	STM32L0x1 (category 5 有效)	有交换功能,但是没有单双切换功能.
	STM32L0x2 (category 5 有效)	有交换功能,但是没有单双切换功能.
	STM32L1xxxx(Cat.4/5/6 devices)	有双 Bank 无需切换
	STM32L4x1(512K)	有单双切换功能,扇区表一样

	STM32L4x5/6(256/512K)	有单双切换功能,扇区表一样
	STM32L4RxxG	有单双切换功能,扇区表 4K/8K,双/单切换(512K 芯片可能已停产)
	STM32L4RxxI	
	STM32L4SxxI	
	ChipSTM32G4 Category3	有单双切换功能,扇区表 2K/4K,双/单切换
	STM32L552xE	有单双切换功能,扇区表 2K/4K,双/单切换
	STM32L562xE	

Power Writer 内置针对双 Bank 芯片做了双分区表，所以用户在使用 Power Writer 时无须关心单双 Bank 的切换或者是擦除方式，由 Power Writer 自动处理数据地址和扇区表的信息，表 7.3 汇总了常用芯片的 Bank 信息汇总，供用户查阅参考

表 7.2 Bank 信息不完全列表参考

6.4Power Writer 保护级别信息汇总

Power Writer 支持完整的 Option Byte，大部分时候用户无须关心芯片是否支持 LEVEL-2，可以直接从 Option Byte 处选择，如果用户在选型芯片时提供快速参考，请查阅表 7.4

芯片厂商	芯片系列	是否支持 Level-0	是否支持 Level-1	是否支持 Level-2
ST	STM32F0xx	V	V	V
	STM32F1xx	V	V	X
	STM32F2xx	V	V	V
	STM32F3xx	V	V	V
	STM32F4xx	V	V	V
	STM32F7xx	V	V	V
	STM32L0xx	V	V	V
	STM32L1xx	V	V	V
	STM32L4xx	V	V	V
	STM32G0xx	V	V	V
	STM32H7xx	V	V	V
GD	GD32F10x	V	V	X
	GD32F1x0	V	V	V
	GD32F20x	V	V	X
	GD32F30x	V	V	X
	GD32F3x0	V	V	V
	GD32F4xx	V	V	V
	GD32E103	V	V	V

	GD32E230	V	V	V
MM32灵动微	MM32F00x	V	V	X
	MM32F0xx	V	V	X
	MM32F1xx	V	V	X
	MM32L0xx	V	V	X
	MM32L362xx	V	V	X
	MM32L373xx	V	V	X
	MM32SPINxx	V	V	X
	MM32W0xx	V	V	X

表 7.4 主流芯片保护级别参考表

“V”表示支持相应功能，“X”表示不支持相应功能

7 ：联系我们

感谢阁下选择创芯工坊的产品，创芯工坊致力于提供最优质的产品和服务，来赢得广大用户的支持和信任，如果您在使用 Power Writer 的过程中遇到任何问题，或者是需要定制化产品，都可以通过以下方式和我们取得联系。

网址：https://www.icworkshop.com
邮箱：cs@icworkshop.com
电话：0755-83831322
地址：深圳市罗湖区翠竹街道田贝一路文锦广场文安中心 1309 室

扫码关注微信公众号

官方网站

创芯工坊硬件平台技术部 出品