Version: Next

2 : Summary

2.1 Parameters

ICWKey Functional parameters

2.2 Installation

ICWKey Software

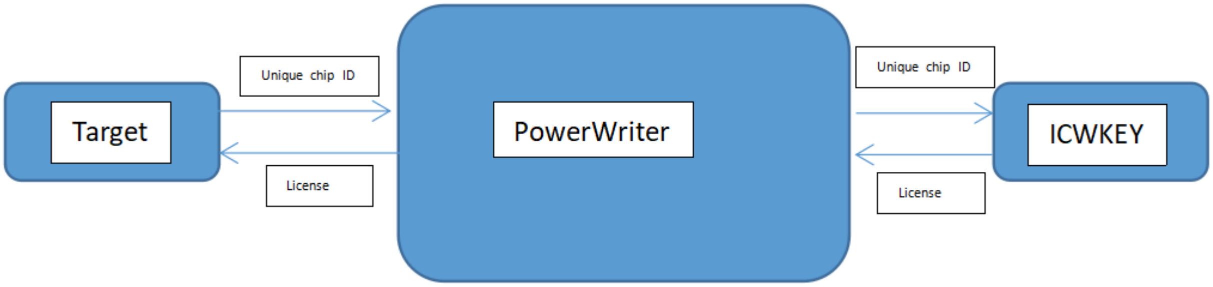
Skip to main content

Version: Next

2.1 Parameters

2.1.1 Summary

ICWorkshop is committed to empowering the traditional IC industry with Internet+ technology to provide a safer and more efficient production management model for the traditional IC production process! In order to meet the needs of users for safe chip programming, authorization control and customized production, ICWorkshop has launched the "Security Authorization Shield (ICWKEY)", hereinafter referred to as ICWKEY, which is an auxiliary tool for the offline authorization of ICWorkshop programmer, PowerWriter, to control the number of authorizations and generate authorization keys, which ensures that the target chip + PowerWriter + authorization key will be generated at the time of production. ICWKEY is an auxiliary tool to control the number of ICWorkshop and generate authorization keys, which can ensure the security of the entire link layer data of target chip + PowerWriter + ICWKEY during production, ensure that the user's firmware is not illegally accessed, and ensure that the user retains the unique authorization control privileges in his hand, to prevent the possibility of unauthorized copies, ICWKEY is completely in the hands of the user, which is safe and reliable, and the following figure shows the workflow diagram:



ICWKEY provides two UID (Unique Chip ID) authorization algorithms, Vector Matrix Encryption (Matrix) and ECDSA Digital Signature, as well as an SDK for users to develop their own custom authorization algorithms to meet different needs. It provides sample programs on how to use UID authorization algorithms on target chips, and also provides ICWKEY.exe, the Windows software of ICWKEY, which allows users to import the randomly generated authorization algorithm source code of ICWKEY.exe into their own programs.

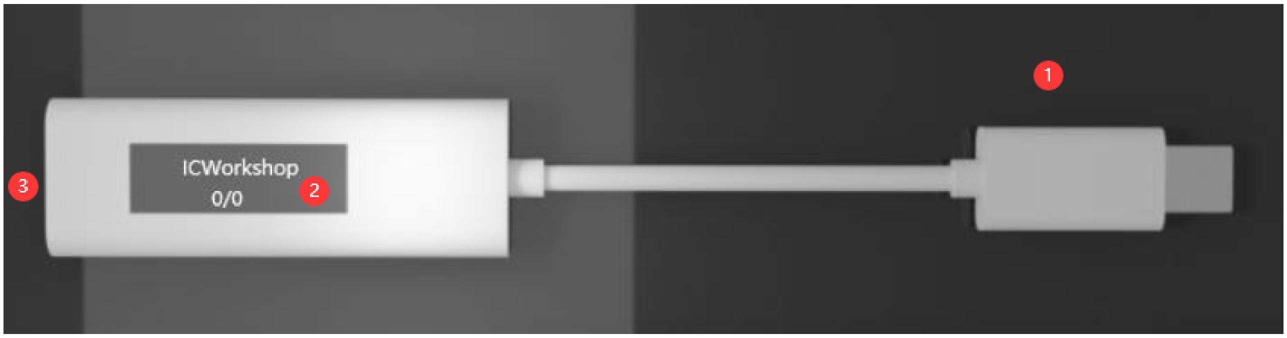
> **TIP**
>
> **MCU General Purpose Advanced Software Protection Library** can provide a higher level of protection, integrated ICWKEY signature, firmware encapsulation, firmware compression, function-level code encryption, firmware validation, object monitoring, unauthorized access detection, debugger detection, privilege separation and control, and other rich security features, to further enhance the security of the software, for more information, please email to cs@icworkshop.com get more detailed information (**not currently publicly available to prevent abuse**).

2.1.2 Product Parameters

- **Size**： 57mm x 22.5mm x 10.6mm (≈)
- **Operating voltage**： 5V (**USB Type-C**)
- **Product Power Consumption**： 60mA~ 90mA

> **TIP**
>
> The parameters of the products are theoretical data, because of the batch, working environment, product improvement and other reasons, there may be differences in the actual, for reference only, subject to change without notice!

2.1.3 Interface



- ①：PowerWriter® Type-C Host Port (connected to PowerWriter)。
- ②：ICWKEY OLED monitor
  - Project name: Displayed in the format: **SafeLic_xxxxxxxx**， xxxxxxxx hash for random item names。
  - Remaining/total : such as 998/1000, the number of available signatures is 998, and the total number of signatures is 1000.
- ③：Type-C slave port (on a computer) ：ICWKEY Powered communication (connected to PC).

2.1.4 Characteristics

☑ Unique ID signature range can be restricted

☑ Number of signatures can be controlled

☑ Configurable number of times (reuse control)

☑ Authorization log query

☑ Signature test

☑ Localizations

- ☑ Signature support
  - ☑ Matrix sign
  - ☑ ECDSA sign

2.1.5 Safety Features

- ICWKEY Developed with a secure chip and integrated with advanced software protection libraries to protect firmware security.
- ICWKEY and PC / ICWKEY and PowerWriter communication encryption, built-in anti-brute force breaking mechanism, can not be cracked by exhaustive password cracking.
- Data dual zone, encrypted design, power down emergency storage.
- Extra long life design.

Edit this page

Skip to main content

Version: Next

2.2 Installation

2.2.1 Summary

The ICWKEY client provides a complete access and configuration interface to the ICWKEY device.

2.2.2 Client Installation

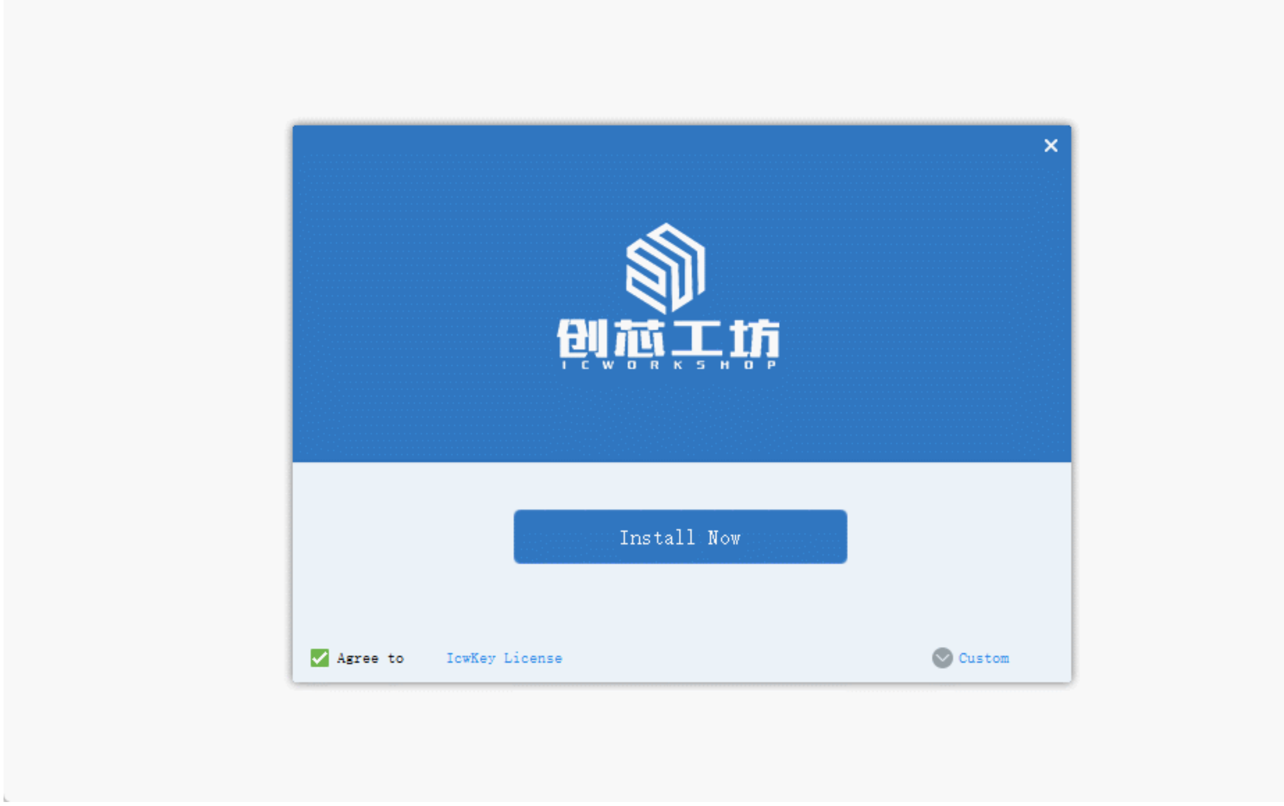
2.2.2.1 Software download address

See the official website for the latest client download address:

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

Please download the ICWKey installation package according to your current system platform.

2.2.2.2 Software Installation Process

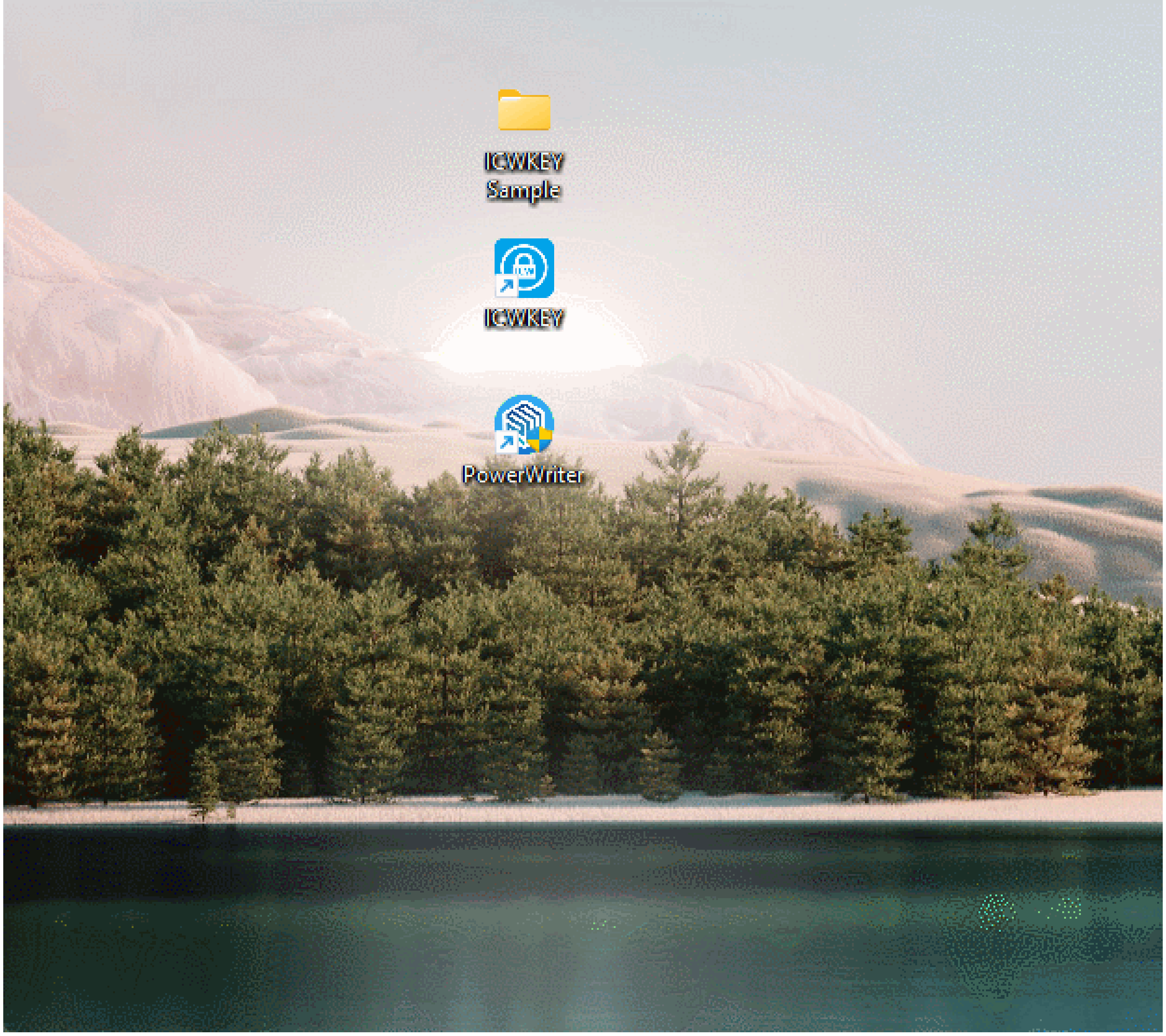


2.2.2.3 Quick start

- Launch ICWKEY from the system desktop by locating the ICWKEY icon.
- Search for ICWKEY from the quick search bar and launch it.

2.2.3 USB Driver Installation

ICWKEY use USB Virtual COM Port to connect to the computer, the first time you connect to the computer, prompted to install the driver, if the computer is Windows 10 system, the system will automatically complete the installation of the driver, if the system is earlier than Windows 10 may need to install manually, the installation package in . \USB_driver\ STSW_STM32102_V1.4.0, read the readme.txt, and then double-click VCP_V1.4.0_Setup.exe to start the installation, the demonstration is as follows:



Edit this page

Skip to main content

Version: Next

3：Quick Start

3.1 PowerWriter Configuration

ICWKey Basic Configuration

3.2 ICWKEY Configuration

ICWKey configuration

3.3 ECDSA Sample

Detailed demonstration of the use of the ECDSA signature algorithm, as well as the caveats

3.4 Matrix Sample

Detailed demonstration of the use of the Matrix signature algorithm, as well as the caveats

Skip to main content

Version: Next

3.1 PowerWriter Configuration

3.1.1 Descriptive

ICWKEY needs to be used with PowerWriter, both must use the same project name and communication key in order to complete the communication, in addition to the PowerWriter side needs to be configured to sign the write address.

3.1.2 Configuration

The configuration process is referenced below:

☑ Open the PowerWriter software and load the existing project or select the chip that needs to be signed to create a new project.

☑ Select Burner Setup Page -> Authorization & Signature -> Please select from the Signature Mode field: **ICWKEY authorizations(or lock mode)**.

☑ Modify **authorization address**: modify the authorization address to the address where the signature information is actually stored(location stored in the firmware, e.g. setting to 0x08002000 means that the signature information needs to be stored at location 0x08002000).

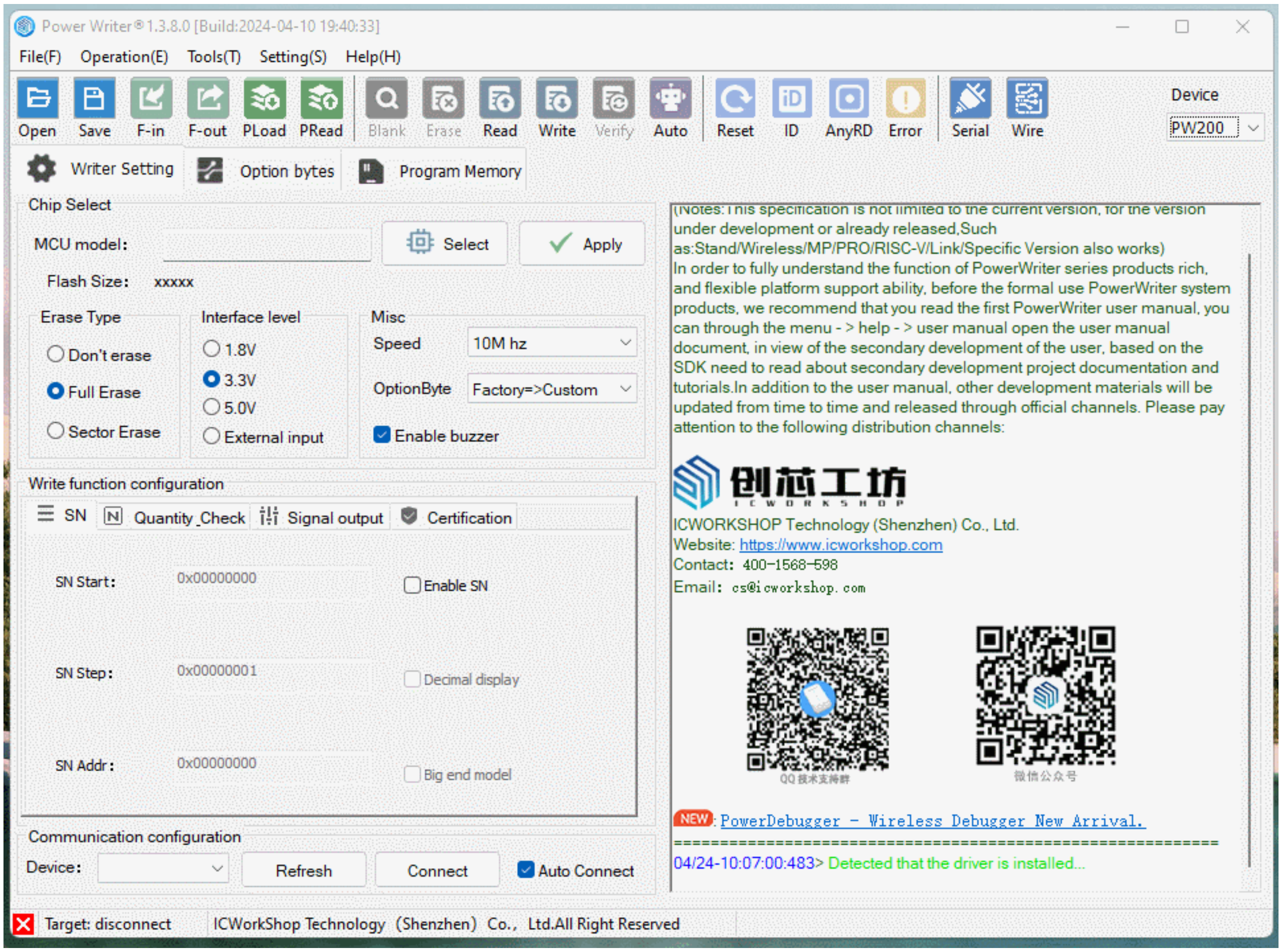
☑ When you are finished setting up, save the PowerWriter project to avoid losing information.

⚠ **CAUTION**

- The authorization address is the address where the signature information is actually stored, which is different for each item. When PowerWriter selects the chip for the first time, the address will be set to the end of the firmware.
- After the setup is complete, save the PowerWriter project so that the configuration information is not lost and the ICWKEY device cannot be connected.
- **Locked Mode** Additional Note: Locked Mode prevents the communication configuration from being viewed and modified again after the next reload of the project (the signature address can be modified).

3.1.3 Demonstrations

A demonstration of the configuration process on the PowerWriter side is shown below.



MCU model:

Flash Size: xxxxx

Erase Type

○ Don't erase

● Full Erase

○ Sector Erase

Interface level

○ 1.8V

● 3.3V

○ 5.0V

○ External input

Misc

Speed 10M hz

OptionByte Factory=>Custom

☑ Enable buzzer

Write function configuration

SN Start: 0x00000000 ☐ Enable SN

SN Step: 0x00000001 ☐ Decimal display

SN Addr: 0x00000000 ☐ Big end model

Communication configuration

Device:

☑ Auto Connect

(Notes:This specification is not limited to the current version, for the version under development or already released,Such as:Stand/Wireless/MP/PRO/RISC-V/Link/Specific Version also works)

In order to fully understand the function of PowerWriter series products rich, and flexible platform support ability, before the formal use PowerWriter system products, we recommend that you read the first PowerWriter user manual, you can through the menu - > help - > user manual open the user manual document, in view of the secondary development of the user, based on the SDK need to read about secondary development project documentation and tutorials.In addition to the user manual, other development materials will be updated from time to time and released through official channels. Please pay attention to the following distribution channels:

创芯工坊 ICWORKSHOP

ICWORKSHOP Technology (Shenzhen) Co., Ltd.
Website: https://www.icworkshop.com
Contact: 400-1568-598
Email: cs@icworkshop.com

QQ 技术支持群

微信公众号

NEW: PowerDebugger - Wireless Debugger New Arrival.

```
==============================================================
04/24-10:07:00:483> Detected that the driver is installed...
```

☒ Target: disconnect

ICWorkShop Technology (Shenzhen) Co., Ltd.All Right Reserved

Edit this page

3 : Quick Start > 3.2 ICWKEY Configuration

Skip to main content

Version: Next

3.2 ICWKEY Configuration

3.2.1 Configuration

The configuration process is referenced below:

☑ Connecting ICWKEY devices (new devices with default communication configuration, reused devices loading previous project files)

☑ Copy the communication password and project name from the PowerWriter configuration side to the configuration side of ICWKEY.

☑ Setting the number of authorizations (which controls how many signatures can actually be performed)

☑ Check and configure the number of times it can be configured (if you want the device to be reused, you don't have to adjust it, if it is used once and then voided, configure it as once)

☑ Select the signature algorithm on the UID Algorithm page, export the algorithm source code, and save it.

☑ Click the Save and Update button to configure the ICWKEY.

☑ Save the ICWKEY project file according to the pop-up window.

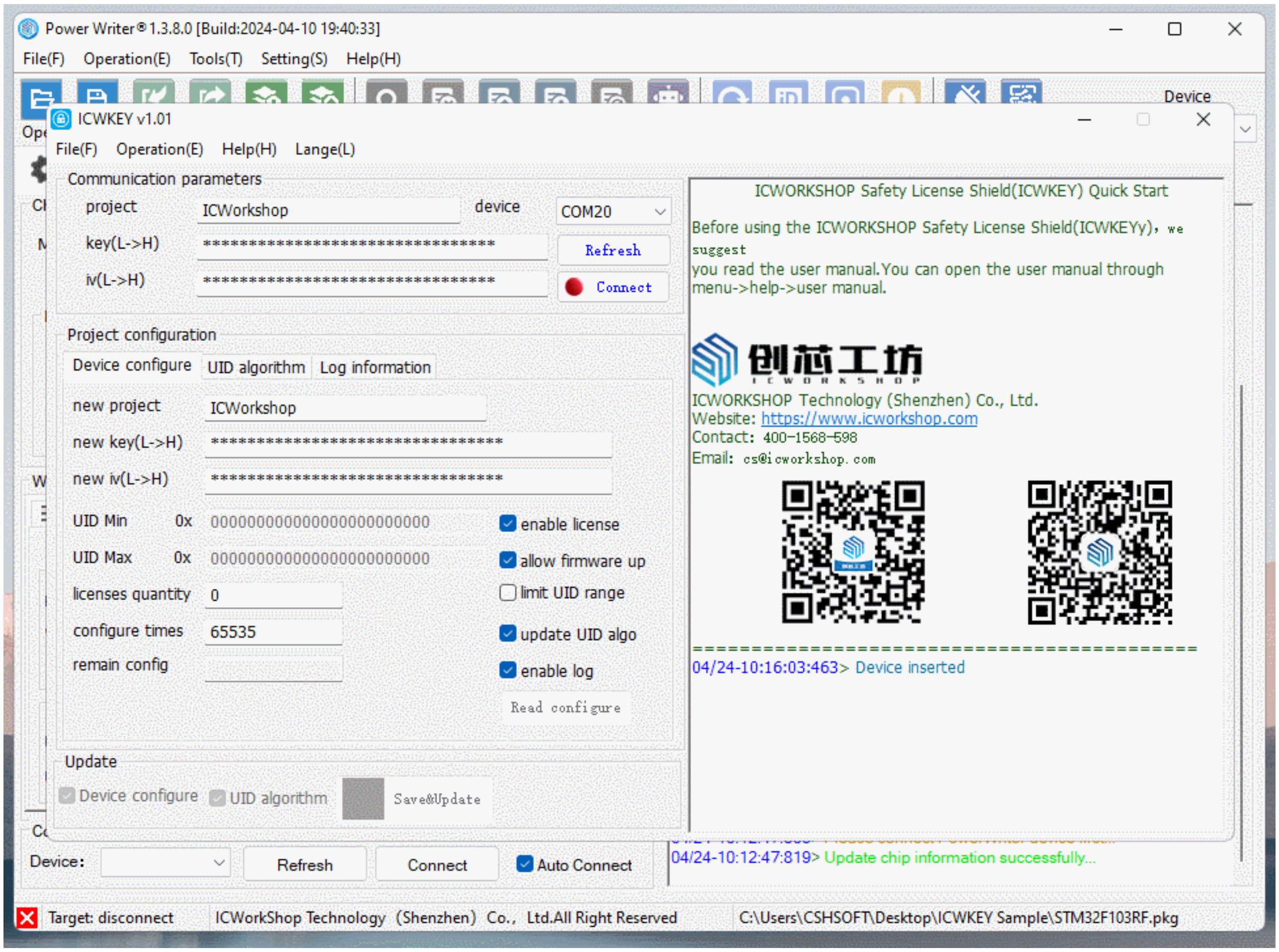
CAUTION

- The communication configuration is generated by the PowerWriter and copied to the ICWKEY's configuration window.
- UID algorithm selection, random generation -> export source code -> save the settings, the subsequent need to do development integration based on the exported source code.

- After ICWKEY is configured, please save the project and remember the project password, losing the project (connection information), you will not be able to connect to the ICWKEY device.
- Configurable number of times Special Note: The default is 65535 reusable times, every time you update the device, the number of times -1, when the number of times reaches 0, at this time, ICWKEY device, will not be able to change any information.

3.2.2 Demonstrations

A demonstration of the configuration flow on the ICWKEY side is shown below.


Power Writer® 1.3.8.0 [Build:2024-04-10 19:40:33]
File(F) Operation(E) Tools(T) Setting(S) Help(H)
Device
ICWKEY v1.01
File(F) Operation(E) Help(H) Lange(L)
Refresh
Connect
Device configure
UID algorithm
Log information
Read configure
Update
Save&Update
Refresh
Connect

Communication parameters

project: ICWorkshop
device: COM20
key(L->H): **********************************
iv(L->H): **********************************

Project configuration

new project: ICWorkshop
new key(L->H): **********************************
new iv(L->H): **********************************
UID Min 0x: 000000000000000000000000
UID Max 0x: 000000000000000000000000
licenses quantity: 0
configure times: 65535
remain config:

☑ enable license
☑ allow firmware up
□ limit UID range
☑ update UID algo
☑ enable log

Update: ☑ Device configure ☑ UID algorithm

ICWORKSHOP Safety License Shield(ICWKEY) Quick Start

Before using the ICWORKSHOP Safety License Shield(ICWKEYy)，we suggest
you read the user manual.You can open the user manual through menu->help->user manual.

ICWORKSHOP Technology (Shenzhen) Co., Ltd.
Website: https://www.icworkshop.com
Contact: 400-1568-598
Email: cs@icworkshop.com

```
===========================================
04/24-10:16:03:463> Device inserted
```

Device: ☑ Auto Connect

```
04/24-10:12:47:819> Update chip information successfully...
```

☒ Target: disconnect ICWorkShop Technology（Shenzhen）Co., Ltd.All Right Reserved C:\Users\CSHSOFT\Desktop\ICWKEY Sample\STM32F103RF.pkg

Edit this page

Version: Next

3.3 ECDSA Sample

3.3.1 Prepare

ECDSA signature is an asymmetric encryption electronic signature method, the private key is stored in the signature device ICWKEY, and the public key is stored in the project firmware. ICWKEY generates the signature information through the ID of the target chip and the current private key, and then writes the signature information to the specified address of the firmware through the PowerWriter, and during the operation of the firmware, it verifies whether the signature information is valid through the public key + ID, thus determining whether the current chip has been validly authorized, and avoiding the firmware from being directly copied. When the firmware is running, the public key + ID will be used to verify whether the current signature information is valid or not, so as to determine whether the current chip has been validly authorized, and to avoid the firmware from being directly copied and used. Before we start, we need to follow the process of verifying that all the preparatory work has been completed.

☑ ICWKEY signatures (or ICWKEY signature lock mode) are used in PowerWriter projects.

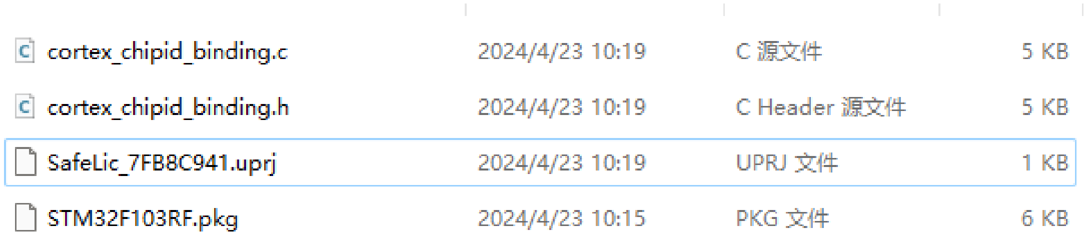
☑ The signature address has been set (e.g. 0x08002000).

☑ The communication information on the PowerWriter side has been synchronized to the project in ICWKEY and encrypted communication with the project has been re-established.

☑ Reasonably set the number of times it can be authorized, for example, set it to 10,000.

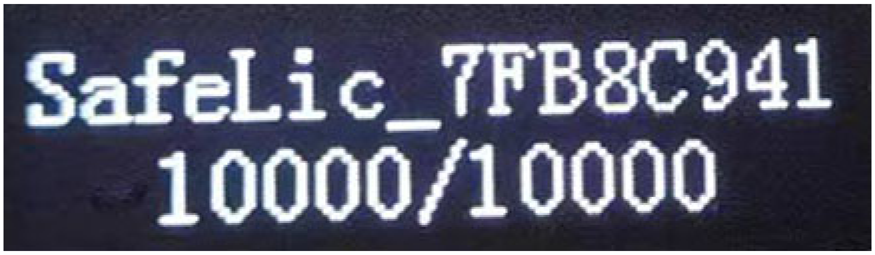
☑ Signature method: ECDSA signature was chosen, saved to ICWKEY, and the source code was exported.

If all the above steps are completed, you can see the display information of ICWKEY device and the exported source code information, refer to the following:



cortex_chipid_binding.c	2024/4/23 10:19	C 源文件	5 KB
cortex_chipid_binding.h	2024/4/23 10:19	C Header 源文件	5 KB
SafeLic_7FB8C941.uprj	2024/4/23 10:19	UPRJ 文件	1 KB
STM32F103RF.pkg	2024/4/23 10:15	PKG 文件	6 KB

At the same time the ICWKEY device will display the following message:



SafeLic_7FB8C941
10000/10000

3.3.2 Sample project

3.3.2.1 Prepare

Sample project path ICWKEY installation path, specifically:

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

ICWKEY desktop icon, you can quickly locate until, and copy the ECDSA sample project to the specified path, and decompression, reference demo as follows:

3.3.2.2 Code structure

3.3.2.2.1 startup_stm32f103xg.s

☑ 调整堆大小 > 0x1300

☑ 挑战栈大小 > 0x800

```
Stack_Size      EQU     0x1000      ;Please make the stack bigger, ECDSA signature verification needs more stack space!

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

CAUTION

Please pay special attention to the stack size, to adjust it, otherwise it will not be able to perform the signature verification and return an out of memory error message.

3.3.2.2.2 cortex_chipid_binding.c

☑ Replace the public key with the one exported by ICWKEY.

```
//Use the public key in ICWKEY for substitution.
const static uint8_t PUBLIC_KEY[49]={

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

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

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

3.3.2.2.3 cortex_chipid_binding.h

☑ Fill in the ID address (see prompt message)

☑ Change the signature address to the signature address in PowerWriter.

☑ Depending on the situation, whether placeholders are turned on or not.

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

> **TIP**
>
> UID_CHIP_ADDR address, you can use PowerWriter to select the signature mode as Matrix, export the source code can see the actual ID address.

3.3.2.2.4 main.c

☑ Initialization ID

☑ Verify Signature

```
/* Private user code
---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
//Used to print log messages from the serial port
int fputc(int ch, FILE *f)
{
        uint8_t ch8 = (uint8_t)ch;
        HAL_UART_Transmit(&huart2,&ch8,sizeof(ch8),5);
        return (ch);
}
/* USER CODE END 0 */

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

  /* USER CODE END 1 */


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

TIP

The sample code is just a demo, for more security, please note that hiding the code can improve the security, if necessary, contact us to get the MCU Common Security Protection Library to further enhance the firmware security and prevent the firmware from being reverse decompiled, cracked, and modified.

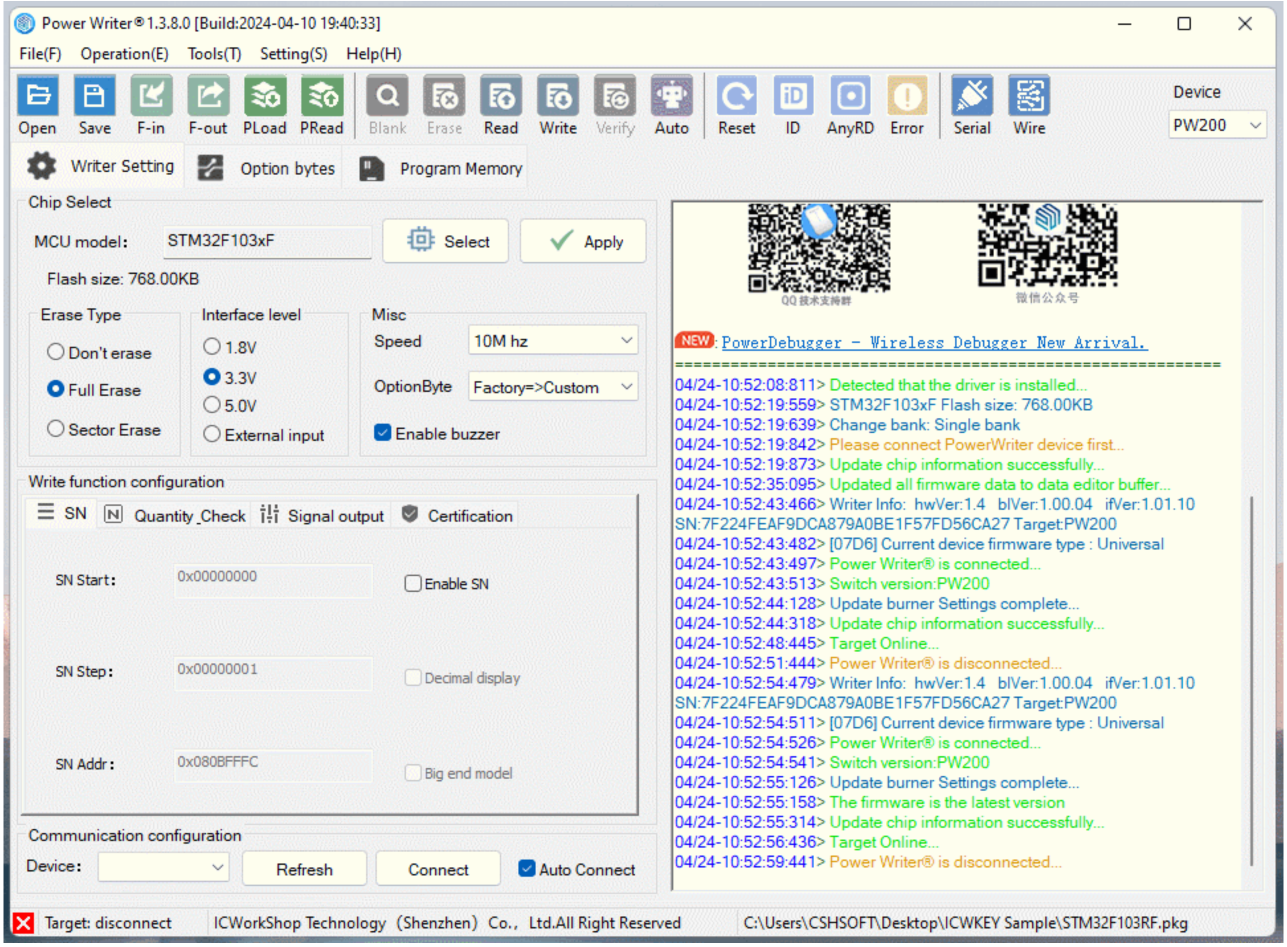
3.3.2.3 Compiling

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

Logging can be turned on during verification for easy viewing of the results. Compile the project and the test firmware will be generated in the directory Output\TargetIC_Example.bin.

3.3.2.4 Validate

Reopen the PowerWriter project, add the TargetIC_Example.bin test firmware to the Program Memory page, and load the project into the PowerWriter device as shown below:



Power Writer®1.3.8.0 [Build:2024-04-10 19:40:33]

Device: PW200

Chip Select

MCU model: STM32F103xF

Flash size: 768.00KB

Erase Type: ○ Don't erase ● Full Erase ○ Sector Erase

Interface level: ○ 1.8V ● 3.3V ○ 5.0V ○ External input

Misc

Speed: 10M hz

OptionByte: Factory=>Custom

☑ Enable buzzer

Write function configuration

SN Start: 0x00000000 □ Enable SN

SN Step: 0x00000001 □ Decimal display

SN Addr: 0x080BFFFC □ Big end model

Communication configuration

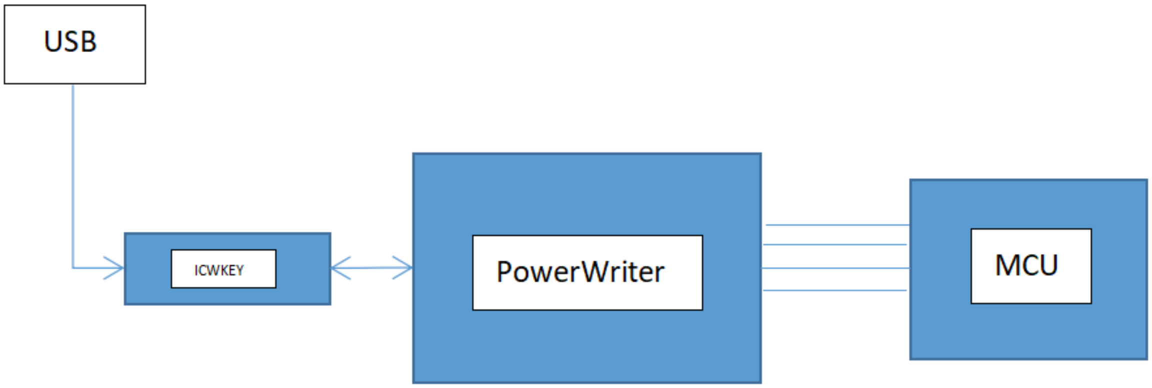
Device:

☑ Auto Connect

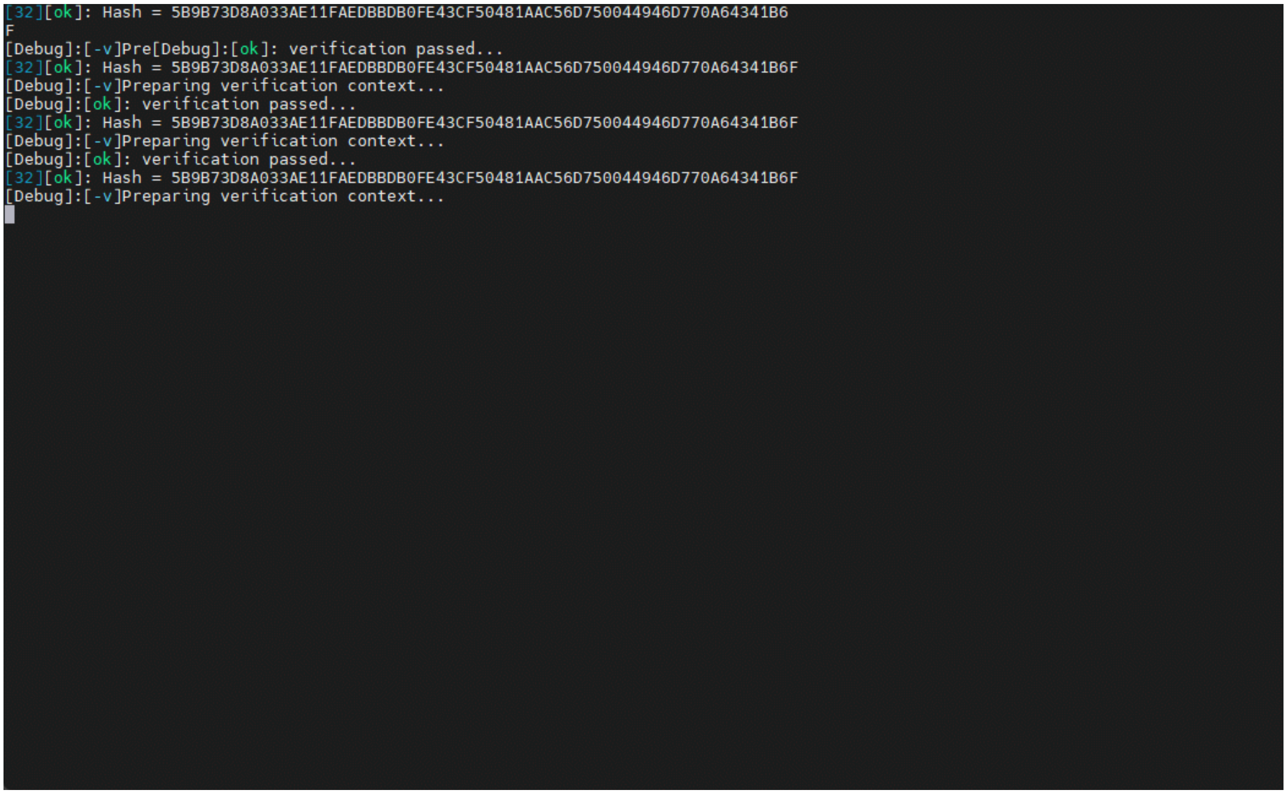
```
NEW: PowerDebugger - Wireless Debugger New Arrival.
=================================================================
04/24-10:52:08:811> Detected that the driver is installed...
04/24-10:52:19:559> STM32F103xF Flash size: 768.00KB
04/24-10:52:19:639> Change bank: Single bank
04/24-10:52:19:842> Please connect PowerWriter device first...
04/24-10:52:19:873> Update chip information successfully...
04/24-10:52:35:095> Updated all firmware data to data editor buffer...
04/24-10:52:43:466> Writer Info:  hwVer:1.4   blVer:1.00.04   ifVer:1.01.10
SN:7F224FEAF9DCA879A0BE1F57FD56CA27 Target:PW200
04/24-10:52:43:482> [07D6] Current device firmware type : Universal
04/24-10:52:43:497> Power Writer® is connected...
04/24-10:52:43:513> Switch version:PW200
04/24-10:52:44:128> Update burner Settings complete...
04/24-10:52:44:318> Update chip information successfully...
04/24-10:52:48:445> Target Online...
04/24-10:52:51:444> Power Writer® is disconnected...
04/24-10:52:54:479> Writer Info:  hwVer:1.4   blVer:1.00.04   ifVer:1.01.10
SN:7F224FEAF9DCA879A0BE1F57FD56CA27 Target:PW200
04/24-10:52:54:511> [07D6] Current device firmware type : Universal
04/24-10:52:54:526> Power Writer® is connected...
04/24-10:52:54:541> Switch version:PW200
04/24-10:52:55:126> Update burner Settings complete...
04/24-10:52:55:158> The firmware is the latest version
04/24-10:52:55:314> Update chip information successfully...
04/24-10:52:56:436> Target Online...
04/24-10:52:59:441> Power Writer® is disconnected...
```

☒ Target: disconnect   ICWorkShop Technology（Shenzhen）Co.，Ltd.All Right Reserved   C:\Users\CSHSOFT\Desktop\ICWKEY Sample\STM32F103RF.pkg

Connect ICWKEY to PowerWriter, and connect the target PCB of MCU to be programmed, and connect the power supply for programming, the reference wiring is shown as follows.



After programming, connect the serial port TX pin of the target PCB, you can see the output signature verification information, refer to the following:



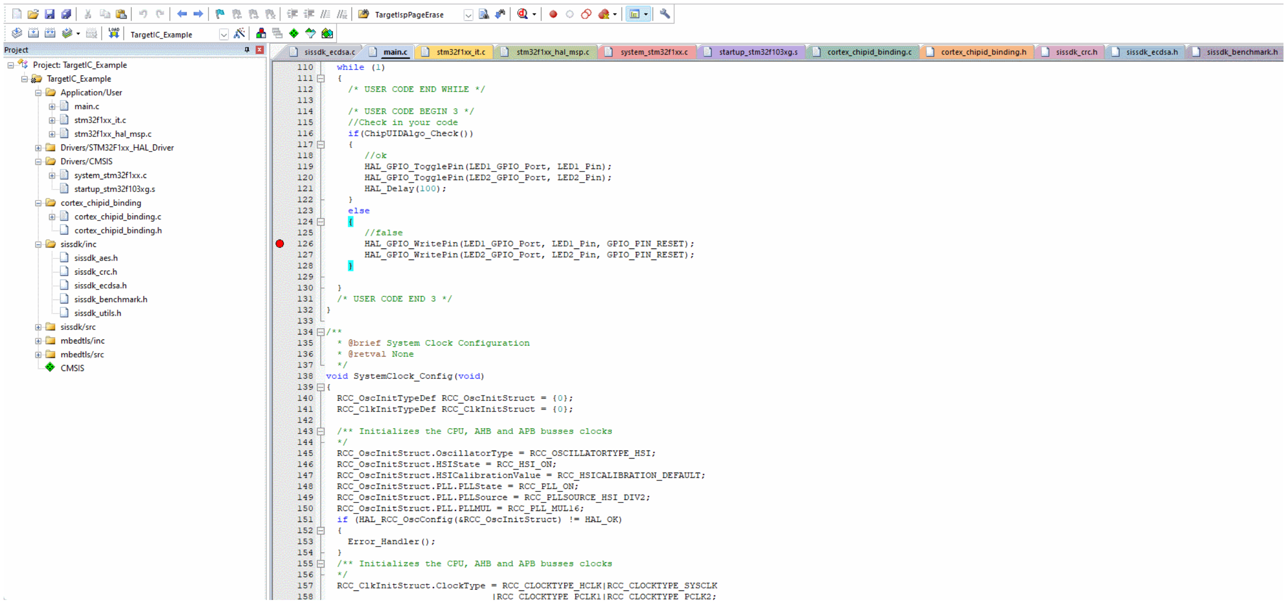
```
[32][ok]: Hash = 5B9B73D8A033AE11FAEDBBDB0FE43CF50481AAC56D750044946D770A64341B6
F
[Debug]:[-v]Pre[Debug]:[ok]: verification passed...
[32][ok]: Hash = 5B9B73D8A033AE11FAEDBBDB0FE43CF50481AAC56D750044946D770A64341B6F
[Debug]:[-v]Preparing verification context...
[Debug]:[ok]: verification passed...
[32][ok]: Hash = 5B9B73D8A033AE11FAEDBBDB0FE43CF50481AAC56D750044946D770A64341B6F
[Debug]:[-v]Preparing verification context...
[Debug]:[ok]: verification passed...
[32][ok]: Hash = 5B9B73D8A033AE11FAEDBBDB0FE43CF50481AAC56D750044946D770A64341B6F
[Debug]:[-v]Preparing verification context...
```

3.3.2.5 Debugging method

After using PowerWriter to sign the target firmware and program it to the target chip, you can check whether the signature is in effect by setting the status output. In complex scenarios, you can't determine the location of the problem simply by looking at the working status, and at this time, you need to debug the target chip, and the debugging steps are as follows:

- Refer to the compilation and verification process to complete the programming
- IDE selection: proceed without erasing the target chip, without programming the target chip, and without verifying the target chip.

The reference demo is shown below:



```
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    //Check in your code
    if(ChipUIDAlgo_Check())
    {
       //ok
       HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
       HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
       HAL_Delay(100);
    }
    else
    {
       //false
       HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
       HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
    }

  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

Edit this page

3 : Quick Start > 3.4 Matrix Sample

Skip to main content

Version: Next

3.4 Matrix Sample

3.4.1 Prepare

Matrix signature is a simple checksum algorithm, ICWKEY (PowerWriter) randomly generates a combination to encrypt the ID, and then write the encrypted information to the target chip during production, the target chip startup, the signature is verified, and is never used to verify that the current chip whether the signature information is written to the method of firmware protection, before starting, we need to follow the process to verify that all preparations have been completed. We need to follow the process to verify that all preparations have been completed.

☑ ICWKEY signatures (or ICWKEY signature lock mode) are used in PowerWriter projects.

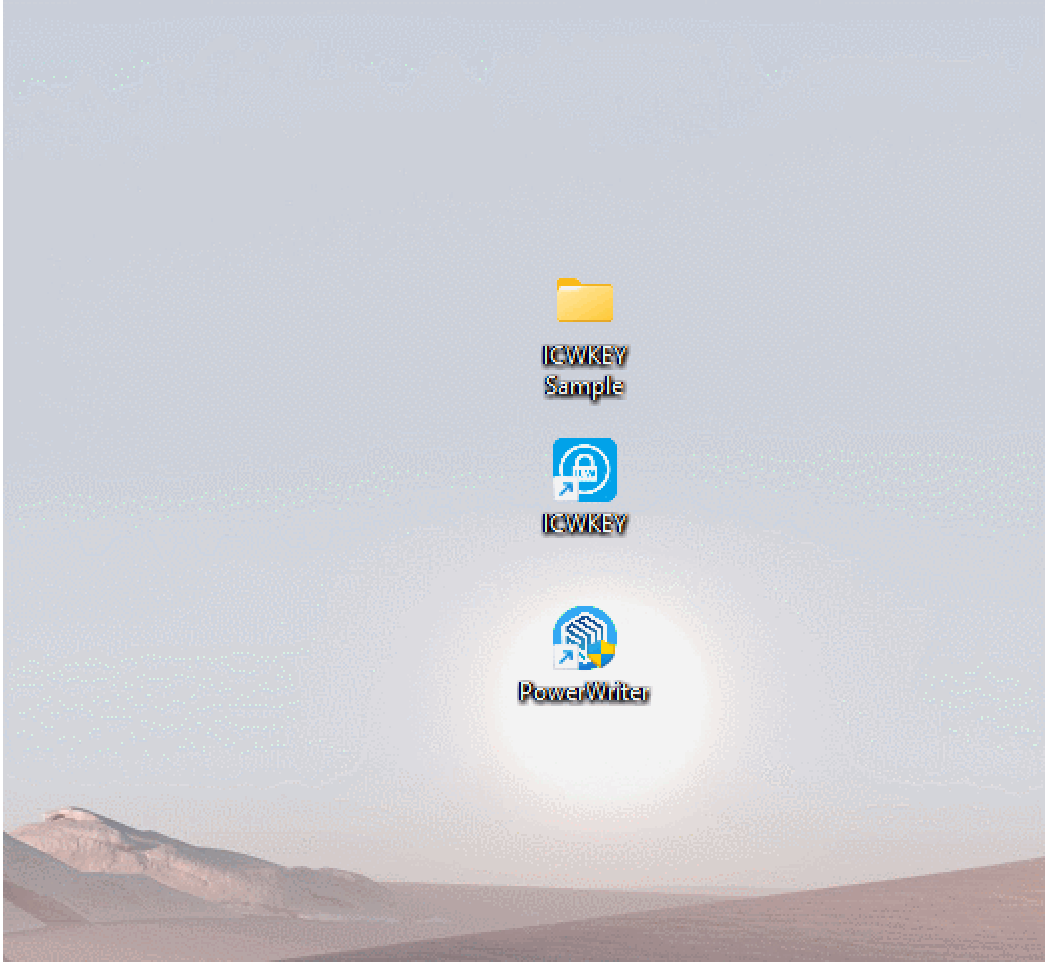
☑ The signature address has been set (e.g. 0x08002000).

☑ The communication information on the PowerWriter side has been synchronized to the project in ICWKEY and encrypted communication with the project has been re-established.

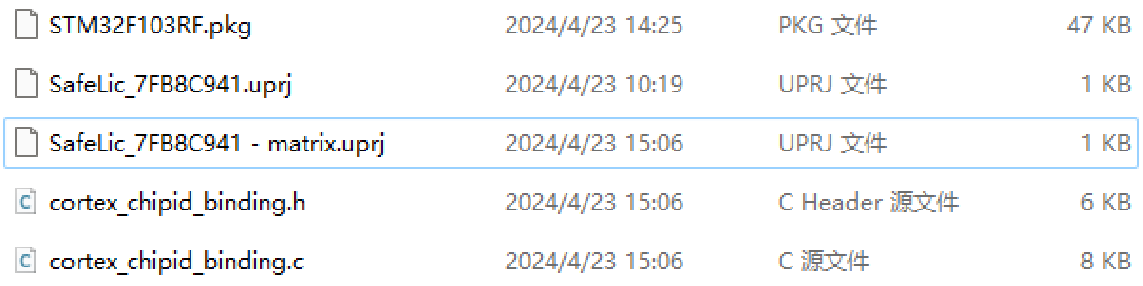
☑ Reasonably set the number of times it can be authorized, for example, set it to 10,000.

☑ Signature method: Matrix signature was chosen, saved to ICWKEY, and the source code was exported.

The demo is shown below:

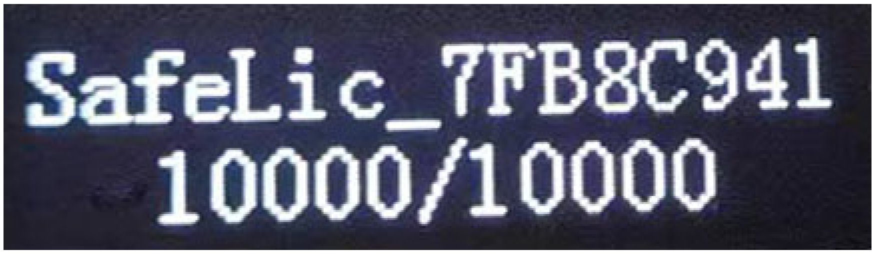


If all the above steps are completed, you can see the display information of ICWKEY device and the exported source code information, refer to the following:



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

At the same time the ICWKEY device will display the following message:



SafeLic_7FB8C941
10000/10000

3.4.2 Sample project

3.4.2.1 Prepare

Sample project path ICWKEY installation path, specifically:

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

ICWKEY desktop icon, you can quickly locate until, and copy the Matrix sample project to the specified path, and decompression, refer to the demo as follows:

3.4.2.2 Code structure

3.4.2.2.1 cortex_chipid_binding.c

☑ Replacing ICWKEY exported functions

```
//Replace it with the exported code from ICWKEY.
//The following code may warn in KEIL(MDK), ignore it
static void ChipUIDAlgo(char pUserID[], char pChipID[], char pKey[])
{
    pKey[0] = pChipID[8] * pChipID[3] | pUserID[8] & pChipID[10] ;
    pKey[1] = pChipID[5] + pChipID[2] - pChipID[7] ^ pChipID[11] ;
```

3.4.2.2.2 cortex_chipid_binding.h

☑ Fill in the ID address (see the prompt).

☑ Change the signature address to the PowerWriter signature address.

☑ Replace with UID_USERID_KEYx exported in ICWKEY .

☑ Depending on the situation, whether placeholders are turned on or not.

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

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

#define    UID_CHIP_MASK                0x5BD489F0              //Random generation
#define    UID_CHIP_SIZE                12                      //ChipID Size
/* ID address of the target chip, which can be queried according to the chip's manual */
#define    UID_CHIP_ADDR                (0x1FFFF7E8^UID_CHIP_MASK) //ChipID Inner Addr in chip

#define    UID_KEY_LENGTH               12                       //The password is the same length as the user ID input
//Signature information storage address, change to the address where the signature information is stored in the PowerWriter project 0x08002000
#define    UID_KEYADDR_INNER            (0x08002000^UID_CHIP_MASK) //Key Store Addr In flash

//Replace with the password exported in ICWKEY.
#define    UID_USERID_LENGTH            UID_KEY_LENGTH            //Customize password length
```

TIP

UID_CHIP_ADDR address, you can use PowerWriter to select the signature mode as Matrix, export the source code can see the actual ID address.

3.4.2.2.3 main.c

☑ Initialization ID

☑ Verify Signature

```
/* Private user code
---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
//Used to print log messages from the serial port
int fputc(int ch, FILE *f)
{
        uint8_t ch8 = (uint8_t)ch;
        HAL_UART_Transmit(&huart2,&ch8,sizeof(ch8),5);
        return (ch);
}
/* USER CODE END 0 */

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

  /* USER CODE END 1 */


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

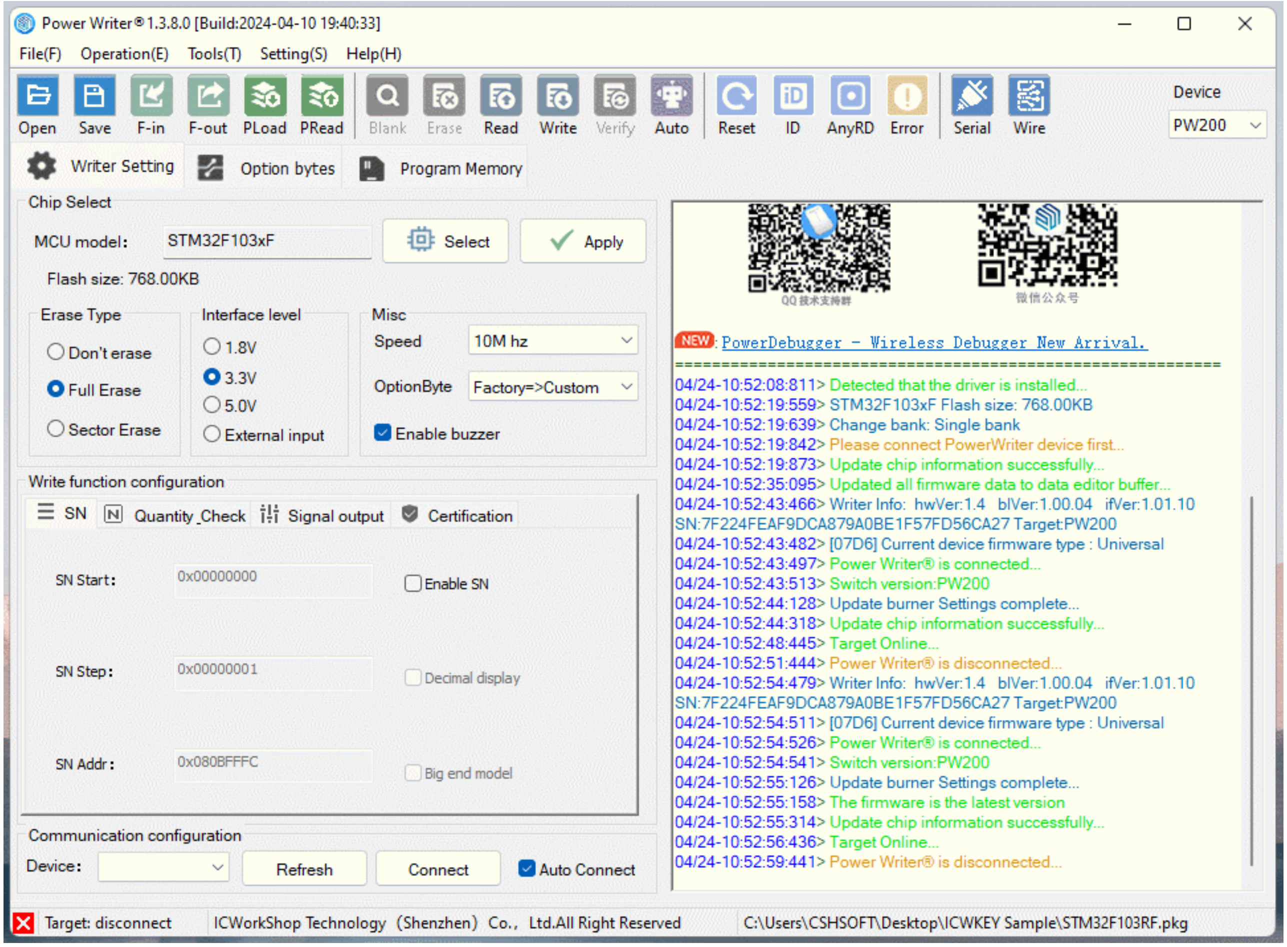
> **TIP**
>
> The sample code is just a demo, for more security, please note that hiding the code can improve the security, if necessary, contact us to get the MCU Common Security Protection Library to further enhance the firmware security and prevent the firmware from being reverse decompiled, cracked, and modified.

3.4.2.3 Compiling

Compiling the project will generate the test firmware in the directory Output\TargetIC_Example.bin.

3.4.2.4 Validate

Reopen the PowerWriter project, add the TargetIC_Example.bin test firmware to the Program Memory page, and load the project into the PowerWriter device as shown below:


Power Writer® 1.3.8.0 [Build:2024-04-10 19:40:33]
File(F) Operation(E) Tools(T) Setting(S) Help(H)
Open Save F-in F-out PLoad PRead Blank Erase Read Write Verify Auto Reset ID AnyRD Error Serial Wire
Writer Setting
Option bytes
Program Memory
Select
Apply
Refresh
Connect
QQ 技术支持群
微信公众号

Device: PW200

Chip Select

MCU model: STM32F103xF

Flash size: 768.00KB

Erase Type: ○ Don't erase ● Full Erase ○ Sector Erase

Interface level: ○ 1.8V ● 3.3V ○ 5.0V ○ External input

Misc

Speed: 10M hz

OptionByte: Factory=>Custom

☑ Enable buzzer

Write function configuration

SN | Quantity_Check | Signal output | Certification

SN Start: 0x00000000 □ Enable SN

SN Step: 0x00000001 □ Decimal display

SN Addr: 0x080BFFFC □ Big end model

Communication configuration

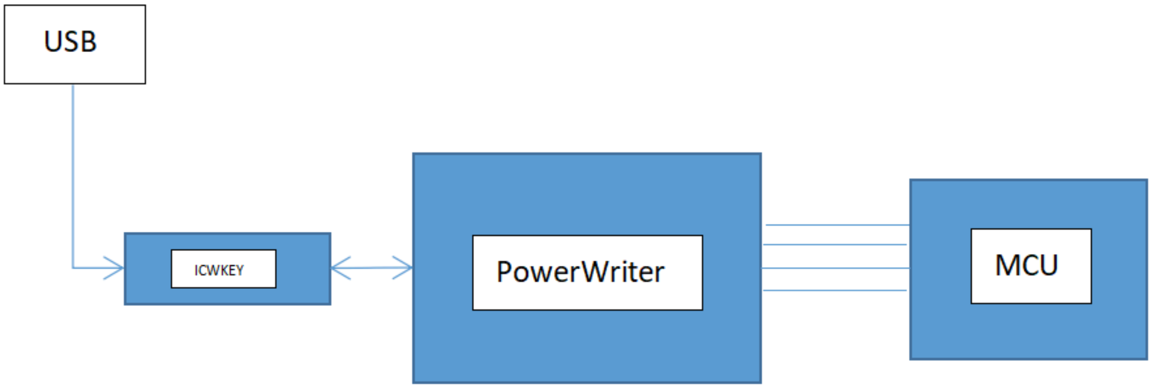
Device: ☑ Auto Connect

NEW: PowerDebugger - Wireless Debugger New Arrival.

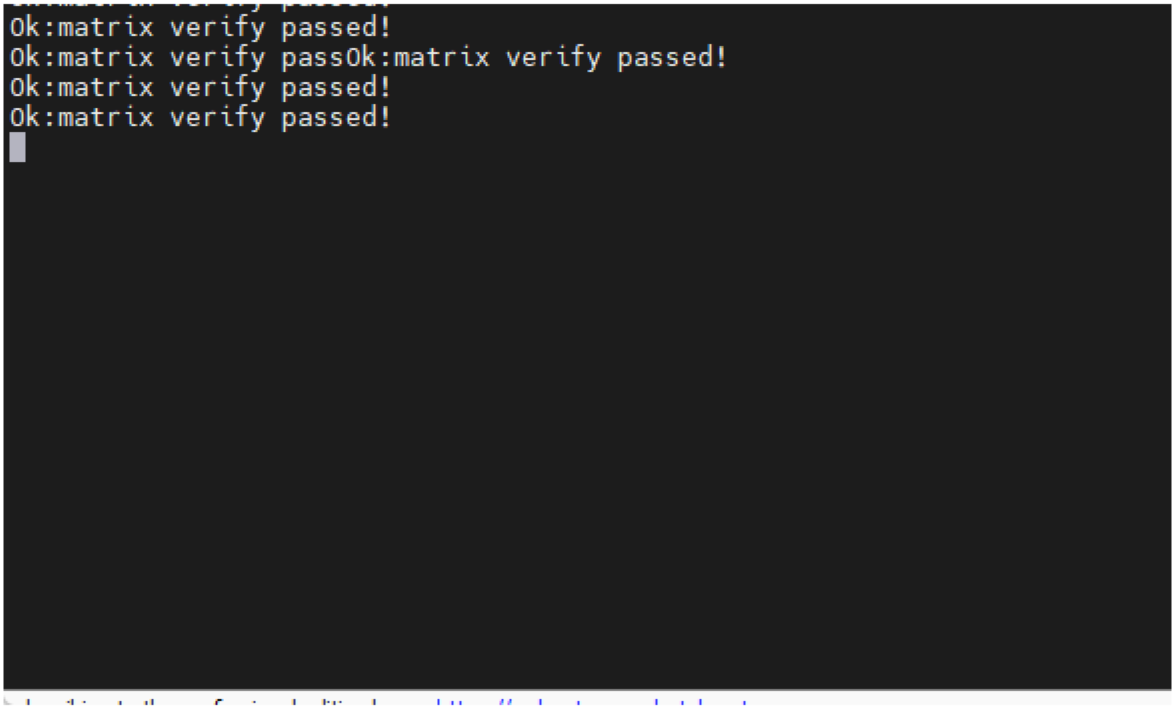
```
==================================================================
04/24-10:52:08:811> Detected that the driver is installed...
04/24-10:52:19:559> STM32F103xF Flash size: 768.00KB
04/24-10:52:19:639> Change bank: Single bank
04/24-10:52:19:842> Please connect PowerWriter device first...
04/24-10:52:19:873> Update chip information successfully...
04/24-10:52:35:095> Updated all firmware data to data editor buffer...
04/24-10:52:43:466> Writer Info:  hwVer:1.4   blVer:1.00.04   ifVer:1.01.10
SN:7F224FEAF9DCA879A0BE1F57FD56CA27 Target:PW200
04/24-10:52:43:482> [07D6] Current device firmware type : Universal
04/24-10:52:43:497> Power Writer® is connected...
04/24-10:52:43:513> Switch version:PW200
04/24-10:52:44:128> Update burner Settings complete...
04/24-10:52:44:318> Update chip information successfully...
04/24-10:52:48:445> Target Online...
04/24-10:52:51:444> Power Writer® is disconnected...
04/24-10:52:54:479> Writer Info:  hwVer:1.4   blVer:1.00.04   ifVer:1.01.10
SN:7F224FEAF9DCA879A0BE1F57FD56CA27 Target:PW200
04/24-10:52:54:511> [07D6] Current device firmware type : Universal
04/24-10:52:54:526> Power Writer® is connected...
04/24-10:52:54:541> Switch version:PW200
04/24-10:52:55:126> Update burner Settings complete...
04/24-10:52:55:158> The firmware is the latest version
04/24-10:52:55:314> Update chip information successfully...
04/24-10:52:56:436> Target Online...
04/24-10:52:59:441> Power Writer® is disconnected...
```

☒ Target: disconnect | ICWorkShop Technology（Shenzhen）Co.，Ltd.All Right Reserved | C:\Users\CSHSOFT\Desktop\ICWKEY Sample\STM32F103RF.pkg

Connect ICWKEY to PowerWriter, and connect the target PCB of MCU to be programmed, and connect the power supply for programming, the reference wiring is shown as follows.



After programming, connect the serial port TX pin of the target PCB, you can see the output signature verification information, refer to the following:



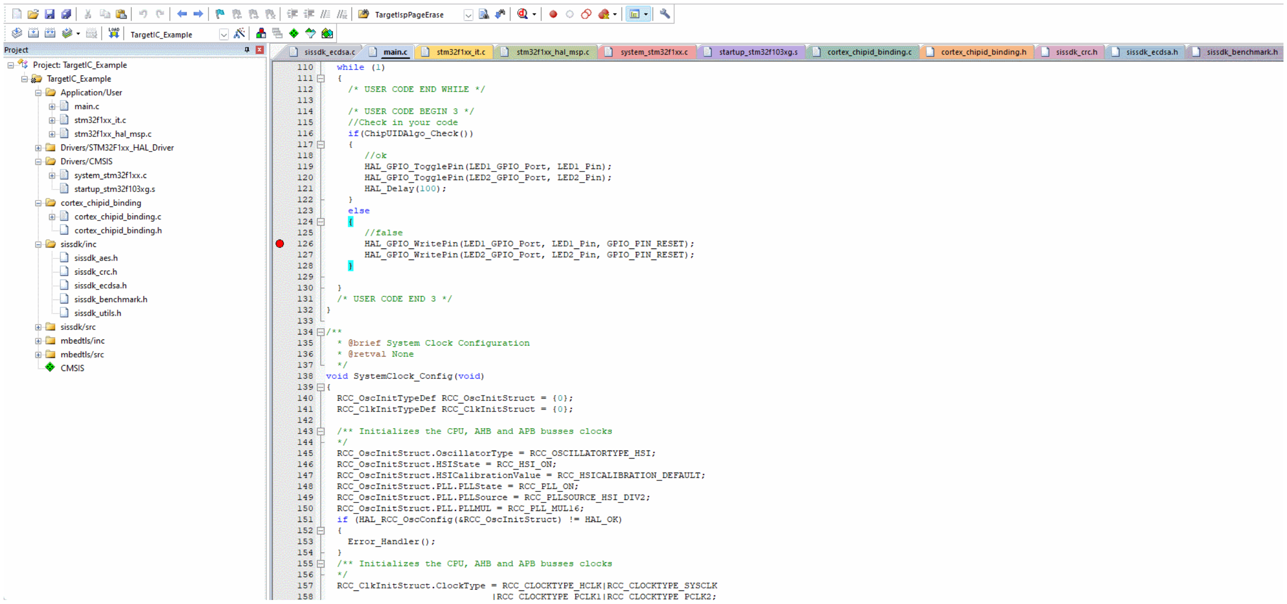
```
Ok:matrix verify passed!
Ok:matrix verify passed!
Ok:matrix verify passOk:matrix verify passed!
Ok:matrix verify passed!
Ok:matrix verify passed!
```

3.4.2.5 Debugging method

After using PowerWriter to sign the target firmware and program it to the target chip, you can check whether the signature is in effect by setting the status output. In complex scenarios, you can't determine the location of the problem simply by looking at the working status, and at this time, you need to debug the target chip, and the debugging steps are as follows:

- Refer to the compilation and verification process to complete the programming
- IDE selection: **No erasing target chip, no programming target chip, no calibrating target chip** is performed.

The reference demo is shown below:



```
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    //Check in your code
    if(ChipUIDAlgo_Check())
    {
       //ok
       HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
       HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
       HAL_Delay(100);
    }
    else
    {
       //false
       HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
       HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
    }

  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

Edit this page

Skip to main content

4 ：Reference Guide

Version: Next

4 ：Reference Guide

Skip to main content

4.1 Menu Functions

4.1.1 File

4.1.1.1 New project

New ICWKEY project, after clicking New Project button, it will reset all the current settings, if you need to save the data, please save the data in advance to avoid data loss.

4.1.1.2 Load Project

Load the ICWKEY project file with the suffix uprj, click Load Project, the Load Project dialog box will pop up, fill in the project password, browse the project path, and then click the OK button, the project will be loaded.

4.1.1.3 Saving Project

Saves the current changes to the project file.

4.1.1.4 Save project as

Save the project as a new project.

4.1.1.5 Exit

Exit the ICWKEY software.

4.1.2 Operation

4.1.2.1 Default Communication Settings

This function allows you to restore the project name, password, vectors, and other information set in the communication settings to the default values as shown below.

```
/*
Default Communication Settings
Project: ICWorkshop
Password: 30313233343536373839414243444546
IV: 46454443424139383736353433323130
*/
```

4.1.2.2 Project communication settings

This function restores the communication connection information to the current settings of the loaded items if the communication settings have been changed to the default communication settings.

4.1.2.3 Save project and update

This function will save the project and synchronize the latest project to the ICWKEY hardware device.

4.1.3 Help

4.1.3.1 Official website

Visit the official website of ICWorkshop and the official website of PowerWriter. www.powerwriter.com.

4.1.3.2 License

View User Agreement.

4.1.3.3 User manual

View ICWKEY user manual offline PDF.

4.1.4 Localization

4.1.4.1 Simplified Chinese

Set to Simplified Chinese.

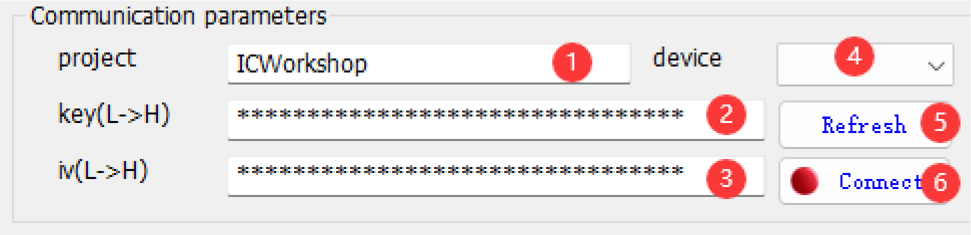
4.1.4.2 English

Set to English.

✎ Edit this page

Version: Next

4.2 Communication Settings



Communication parameters

project ICWorkshop

key(L->H) ********************************

iv(L->H) ********************************

device

- **Project name**：Default as ICWorkshop。
- **Key**：Default as 30313233343536373839414243444546。
- **IV**：Default as 46454443424139383736353433323130。
- **Device List**：Currently recognized to the ICWKEY device list.
- **Refresh**：Refresh the device list.
- **Connect**：Connect the selected device.

Please refer to Demo->Synchronize project name to ICWKEY.

Edit this page

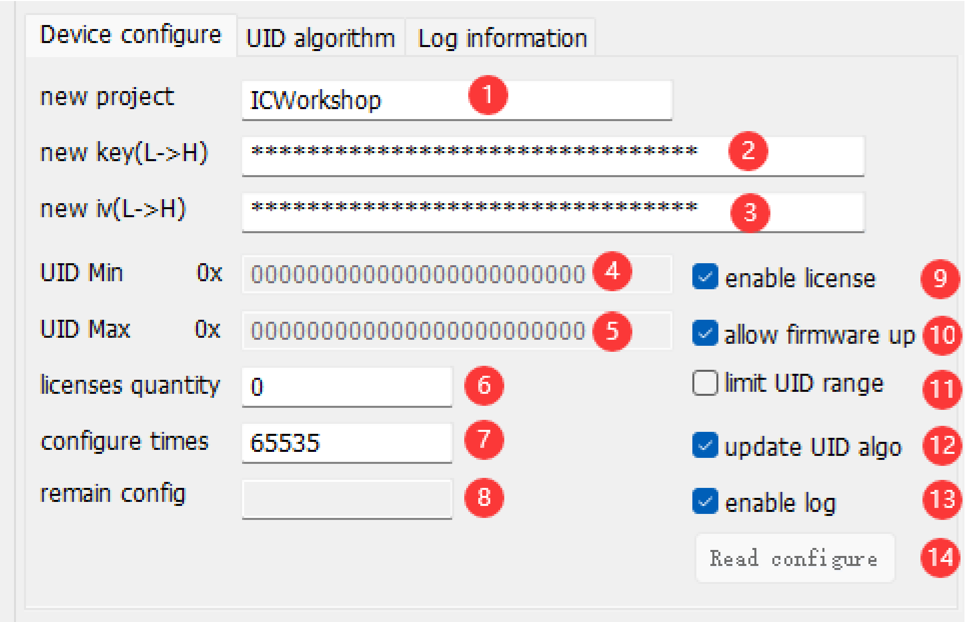
Skip to main content

Version: Next

4.3 Project Configuration

4.3.1 Configuration

The Device Configuration page, which contains most of the configuration needed for signature setup, see the detailed labeled information as shown below:



new project ICWorkshop 1

new key(L->H) ******************************** 2

new iv(L->H) ******************************** 3

UID Min 0x 000000000000000000000000 4

UID Max 0x 000000000000000000000000 5

licenses quantity 0 6

configure times 65535 7

remain config 8

☑ enable license 9

☑ allow firmware up 10

□ limit UID range 11

☑ update UID algo 12

☑ enable log 13

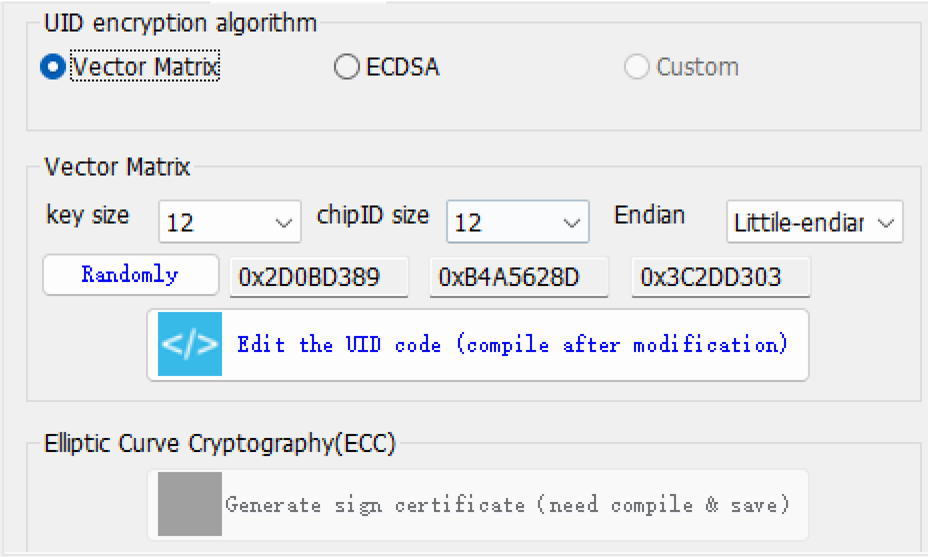
- **New project :** New project name, this field is copied from the PowerWriter project.
- **New Key**：New Password, this field is copied from the PowerWriter project.
- **New IV**：New vector, this field is copied from the PowerWriter project.
- **UID Min**：Limit UID Min.
- **UID Max**：Limit UID maximum.
- **Licenses quantity:** Controls the number of actual authorizations available.

- **Configure times:** The current number of times the ICWKEY device can change the configuration, the default is 65536, every time you configure, the counter -1, when it is 0, it will no longer be able to make changes!
- **Remain config:** The number of modifications remaining for the current device.
- **Enable Authorization**：Switch authorization enable / disable .
- **Allow firmware upgrades**：Whether to allow firmware upgrades.
- **Limit the UID authorization range:** Limit the use of UIDs.
- **Allows updating the UID algorithm:** Whether or not to allow the authorization algorithm to be updated.
- **Enable log:** Records authorisation log information, often used for reports.
- **Read the target configuration:** Reads the configuration information of the current device.

> ⚠ **CAUTION**
>
> - After the new project name, password, and vectors are copied and updated from the PowerWriter project, please save the project to avoid losing it, or else ICWKEY will not be able to connect.
> - UID setting, valid only when Limit UID Authorization Scope is turned on.
> - **Configurable number of times:** Please note that this position is not an authorized number, but the number of times the device can change the settings, when the number of times is 0, the device will be locked, and can not be repeated to modify, unless necessary, do not modify this information.

4.3.2 UID

Currently comes with two signature algorithms, the first is Matrix Signature, a random matrix encryption algorithm, the advantage is that it takes up very little resources, and it can sign and verify the target chip to prevent the firmware from being directly copied and used, and the second is ECDSA Digital Signature, a non-stacked electronic signature algorithm, which is currently a very strong encryption algorithm, and the algorithm is difficult to crack, but it is still necessary to further enhance the protection of the code itself to prevent the signature from being removed. However, it is still necessary to further enhance the protection of the code itself to prevent the signature from being removed, and can be integrated with the MCU Common Advanced Software Protection Library to enhance the security of the firmware, please contact us cs@icworkshop.com。



UID encryption algorithm

● Vector Matrix ○ ECDSA ○ Custom

Vector Matrix

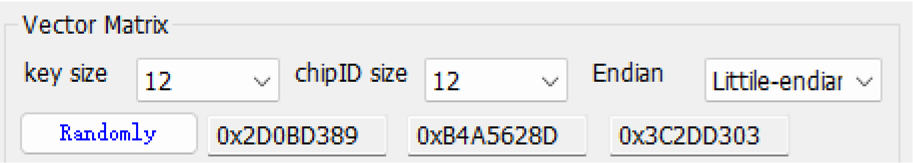
key size: 12 chipID size: 12 Endian: Littile-endiar

0x2D0BD389 0xB4A5628D 0x3C2DD303

Elliptic Curve Cryptography(ECC)

Randomly
Edit the UID code (compile after modification)
Generate sign certificate (need compile & save)

4.3.2.1 Matrix

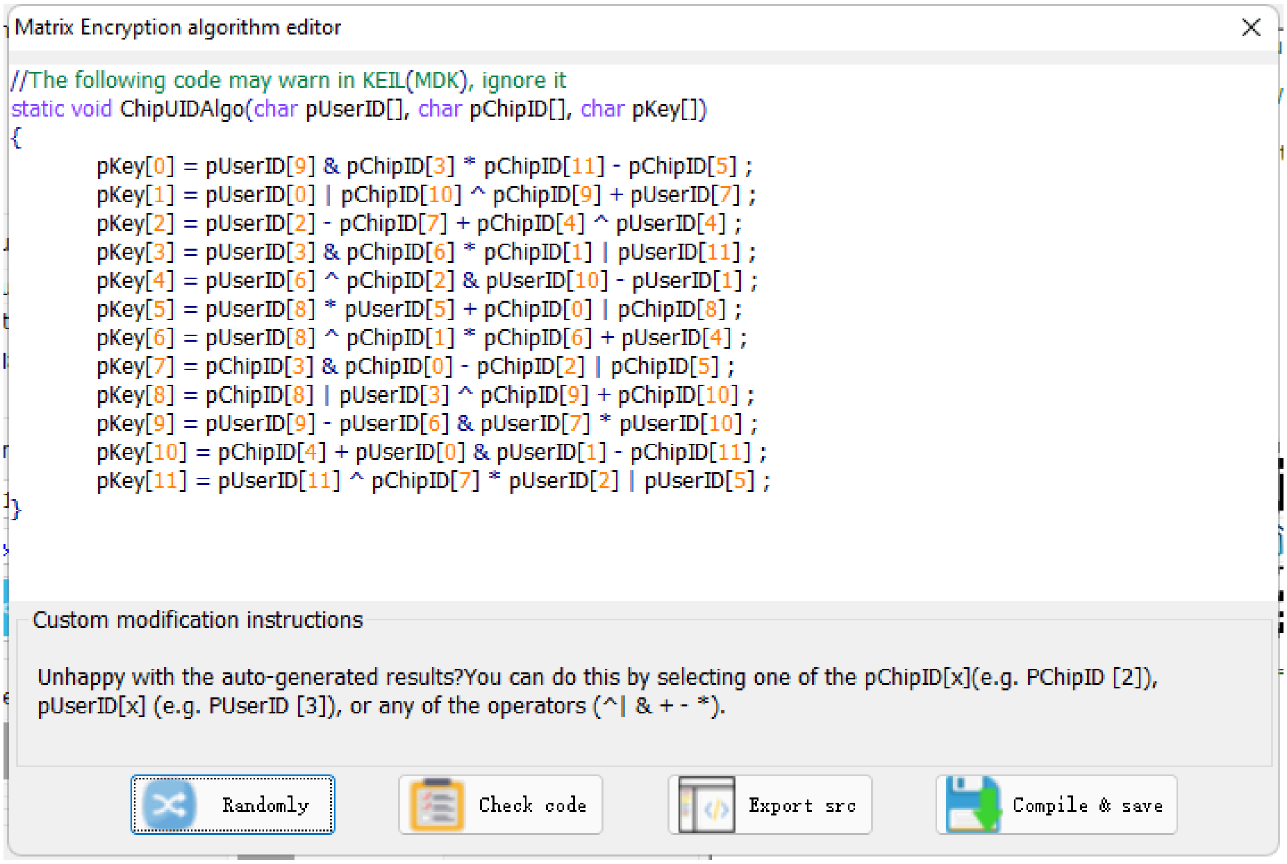


Vector Matrix

key size: 12 chipID size: 12 Endian: Littile-endiar

0x2D0BD389 0xB4A5628D 0x3C2DD303

This setting is generally sufficient to keep the default settings, which can randomly generate Keys.



Matrix Encryption algorithm editor

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

Custom modification instructions

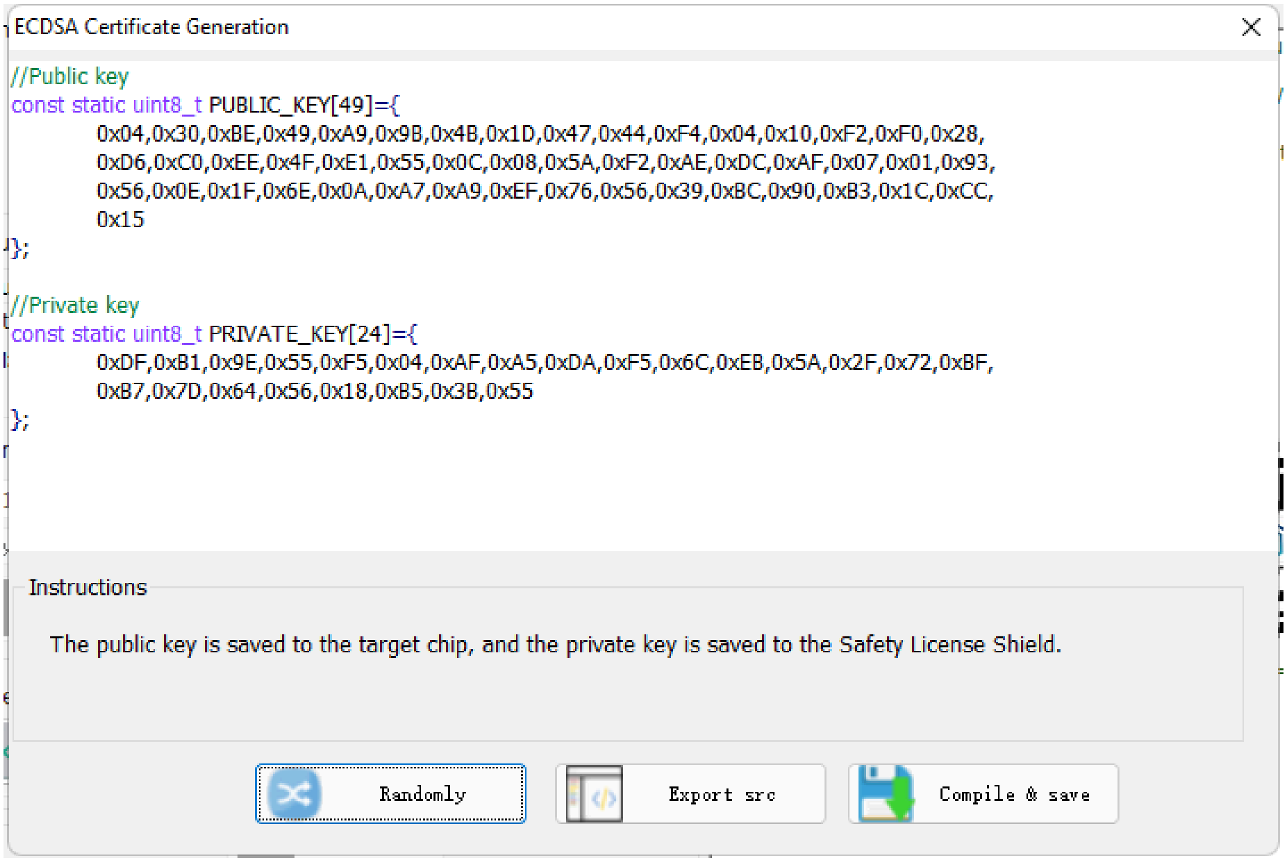
Unhappy with the auto-generated results?You can do this by selecting one of the pChipID[x](e.g. PChipID [2]), pUserID[x] (e.g. PUserID [3]), or any of the operators (^| & + - *).

Randomly generate the Matrix information, when the operation is complete, click Export Source (to be integrated into the firmware), and finally click Compile and Save to update the settings.

TIP

After setting, please be sure to **click the save button** to avoid the setting is not updated, if you forget to operate, you can perform the operation again and export the source code.

4.3.2.1 ECDSA



```
//Public key
const static uint8_t PUBLIC_KEY[49]={
        0x04,0x30,0xBE,0x49,0xA9,0x9B,0x4B,0x1D,0x47,0x44,0xF4,0x04,0x10,0xF2,0xF0,0x28,
        0xD6,0xC0,0xEE,0x4F,0xE1,0x55,0x0C,0x08,0x5A,0xF2,0xAE,0xDC,0xAF,0x07,0x01,0x93,
        0x56,0x0E,0x1F,0x6E,0x0A,0xA7,0xA9,0xEF,0x76,0x56,0x39,0xBC,0x90,0xB3,0x1C,0xCC,
        0x15
};

//Private key
const static uint8_t PRIVATE_KEY[24]={
        0xDF,0xB1,0x9E,0x55,0xF5,0x04,0xAF,0xA5,0xDA,0xF5,0x6C,0xEB,0x5A,0x2F,0x72,0xBF,
        0xB7,0x7D,0x64,0x56,0x18,0xB5,0x3B,0x55
};
```

Instructions

The public key is saved to the target chip, and the private key is saved to the Safety License Shield.

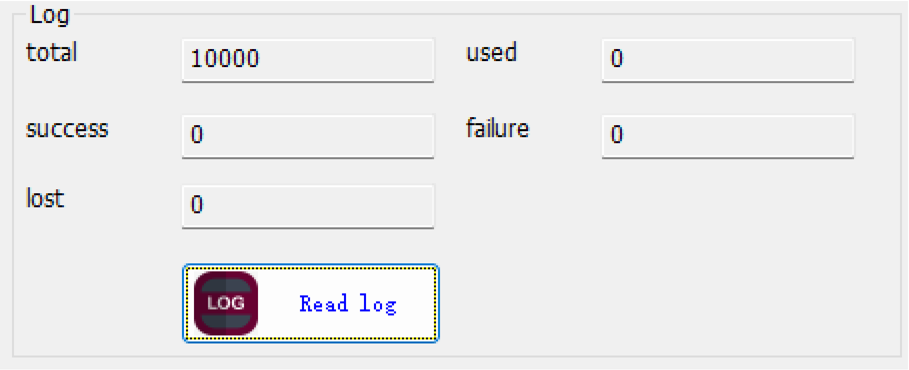
Random Generation: Generates ECDSA digitally signed public-private key pairs.

Export source code: Export source code information for integration.

Compile and Save: After the operation is completed, the settings are saved and updated.

> **TIP**
>
> After setting, please be sure to **click the save button** to avoid the setting is not updated, if you forget to operate, you can perform the operation again and export the source code.

4.3.3 Logger



Log

total 10000

used 0

success 0

failure 0

lost 0

Total number of authorizations: Total number of authorizations currently configured for the device.

Used: Accumulation of the number of times authorization is currently requested.

Number of successes: Accumulation of the number of times a certificate has been successfully distributed.

Number of failures: Total number of distribution failures.

Unknown Error: Unknown error, generally categorized as the number of failures.

Testing authorization:

Filling in the ID information will generate the authorization information of the current chip, which is commonly used for debugging.

TIP

- Please be aware that the signature information returned by ECDSA authorization test may not be the same every time.
- Number of failures + number of successes + unknown errors = number of times

> used.

✎ Edit this page

Version: Next

4.4 Logger

The ICWKEY log field displays the current flow of the operation and the result of the action in the following format

```
    ICWORKSHOP Safety License Shield(ICWKEY) Quick Start

Before using the ICWORKSHOP Safety License Shield(ICWKEYy), we suggest
you read the user manual.You can open the user manual through
menu->help->user manual.



ICWORKSHOP Technology (Shenzhen) Co., Ltd.
Website: https://www.icworkshop.com
Contact: 400-1568-598
Email: cs@icworkshop.com

===========================================
04/24-13:51:45:286> Device inserted
04/24-13:51:54:254> Load successfully:SafeLic_7FB8C941_2.uprj
04/24-13:51:55:023> Start device pairing
04/24-13:51:55:193> Pairing success
04/24-13:51:55:224> Successfully read target device configuration,
                    Project name:SafeLic_7FB8C941,serial
number:21B53974DE21BA1179EF54CA853E89DE,
                    HW version :v1.00,FW version :v1.03,Total
license:10000,Left
                    license:10000,configurable times:65522,uid algorithm
: Elliptic Curve Cryptography
```

Edit this page

Skip to main content

Version: Next

5：Appendix

Skip to main content

5.1 FAQ

1：USB Driver installation failed

If the computer operating system is Windows XP, Win7 or Win8, and is not the official original version, but a lite system, you may encounter installation failure problems, you can search for "ST Virtual Serial Driver Installation Failure" to get a solution, do not select the actual system does not match the driver to install, if necessary, contact technical support. Do not choose a driver that does not match your actual system, and contact technical support if necessary.

2：ICWKEY connection judgment

Ensure that the ICWKEY Client and ICWKEY are disconnected (not paired), ICWKEY is plugged into the PowerWriter's USB socket, and the buzzer will beep twice to indicate a successful connection.

3：Validating Signed Data

When the PowerWriter burns successfully, the green indicator light will be on. If the user is worried that the burned data is not the same as expected, in case the chip does not have read protection turned on, the user can use PowerWriter.exe or other tools to verify the target chip, or read back the data to observe if the data is the same as the original file.

4：Failed during debugging

The signature information is programmed by PowerWriter, and there is no authorization data during debugging, so the checksum cannot be passed, please refer to the ECDSA Sample & Matrix Sample。

5：Signature data length

- Matrix : The length is normally 12 bytes, follow the setting, you can set 4 bytes, 8 bytes, 12 bytes.
- ECDSA: Not more than 141 bytes in length.

6：Programming Failure Reasons

- Wiring problems: wrong wiring, loose wiring.
- Configuration error: The selected chip does not match the target chip.
- Count Exhaustion: The count set by PowerWriter or ICWKEY is exhausted.
- Disable burn-in: The target chip has turned off the burn-in function, for example, the secondary read protection is turned on, and JTAG &serial wire has been disabled.
- Pin reuse: the burn IO is changed to normal IO by the program, try to connect the RESET pin and use the under reset mode to burn.
- Insufficient power supply: Low voltage may cause a burn failure.

7：Cannot reprogram after programming

It may be that the target chip has read protection turned on.

Edit this page

Version: Next

5.2 Notices

1：Source code modification

The following information is usually required to be modified in ECDSA signature mode:

- Replace the public key with the one exported by ICWKEY.
- Fill in the ID address (see the prompted message).
- Modify the signature address to the signature address in the PowerWriter.
- Depending on the situation, whether placeholders are turned on or not.

The following information is usually required to be modified in ECDSA signature mode:

- Replace the function exported by ICWKEY.
- Fill in the ID address.
- Change the signature address to the PowerWriter signature address.
- Replace with UID_USERID_KEYx exported in ICWKEY .
- Depending on the situation, whether placeholders are turned on or not.

Please refer to ECDSA Sample & Matrix Sample。

2：Project password

Keep in mind to save the project files for the PowerWriter project and the ICWKEY. Loss of the project files may result in the inability to properly configure the signature or connect to the ICWKEY device.

3：Signature address

Please don't store the signature address beyond the space of Flash, to avoid not being able to burn, at the same time, please don't overlap with the code, if you are worried about the overlap, please turn on the placeholder, after turning on the placeholder, it will reserve the space in the firmware, so as to avoid overlapping phenomenon, and at the same time, please put the signature address in the front of the address as far as possible.

3：Placeholder

When turned on, space is pre-allocated in the firmware to avoid overwriting firmware data. When not turned on, the data at the specified signature address is overwritten, and the length of the overwrite is referenced to

Signature data length

✎ Edit this page