摘要

本技术文档旨在帮助客户实现 IAP 升级提供一个参考方案和模板。本文档提供基于 UART 升级的 MCU 参考代码和 PC 端的升级工具软件。**SWD 口禁用后， MCU 就不能再通过 SWD 接口烧录更新或者擦除 Flash，如果需要禁用 SWD 样品，后续又需要更新程序的应用场景下，建议用户采用 IAP 更新程序的方案。**

适用 MCU 型号：CS32L010
关键词：　　IAP、Bootloader、 OTA、在线升级、空中升级、串口升级

版本

历史版本	修改内容	日期
V1.0	初版生成	2022-07-07
V1.1	增加 SWD 口禁用说明	2022-07-22

目录

1 MCU Flash Memory 空间分配

图 1 IAP Flash 空间分配说明
Boot 区域占用 3KB 空间：0x0000 0000 - 0x0000 1FFF (8KB 空间)
用户程序占用 29KB 空间：0x0000 2000 - 0x0000 FFFF (56KB 空间)

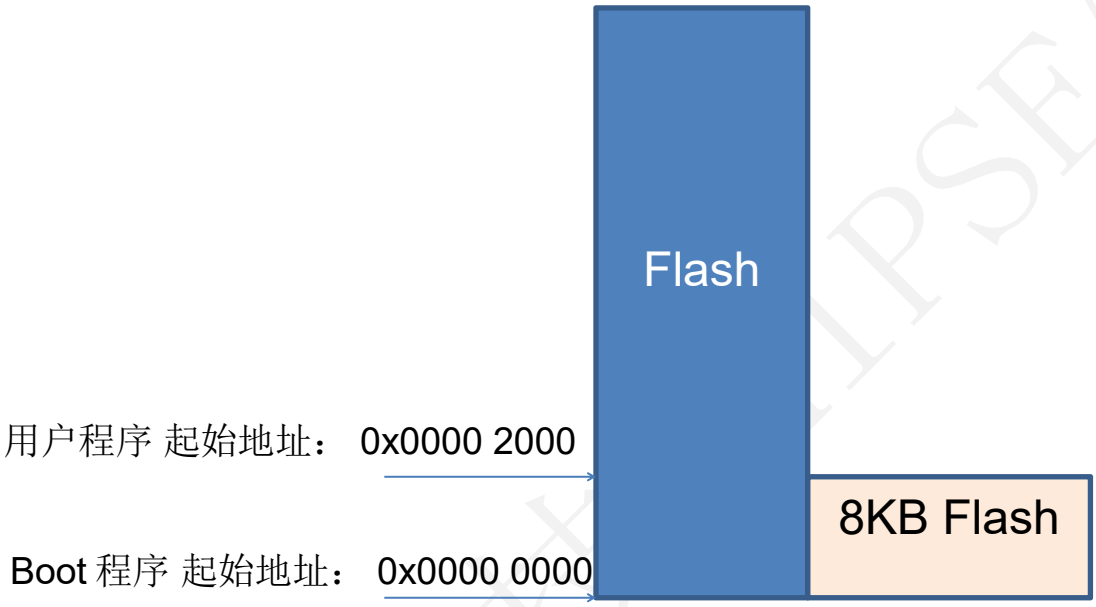


图 1.1

Flash 地址范围	大小	区域说明
0x0000 0000 - 0x0000 1FFF	8KB	Boot区域占用空间
0x0000 0000 - 0x0000 FFFF	56KB	用户程序占用空间

图 1.2

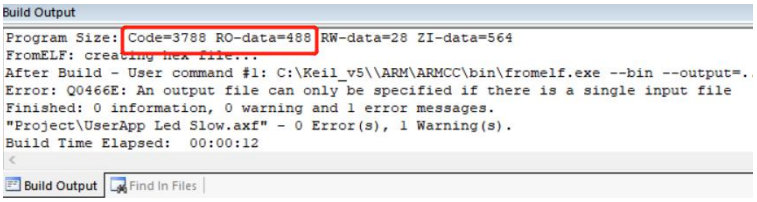


图 1.32 工程编译后，大概占用 4KB 的空间

2 硬件介绍

- CS32L010 开发板
- 串口工具
- 上位机 工具
- Jlink 调试器

  说明 ： IAP 例程基于 KEIL 工程

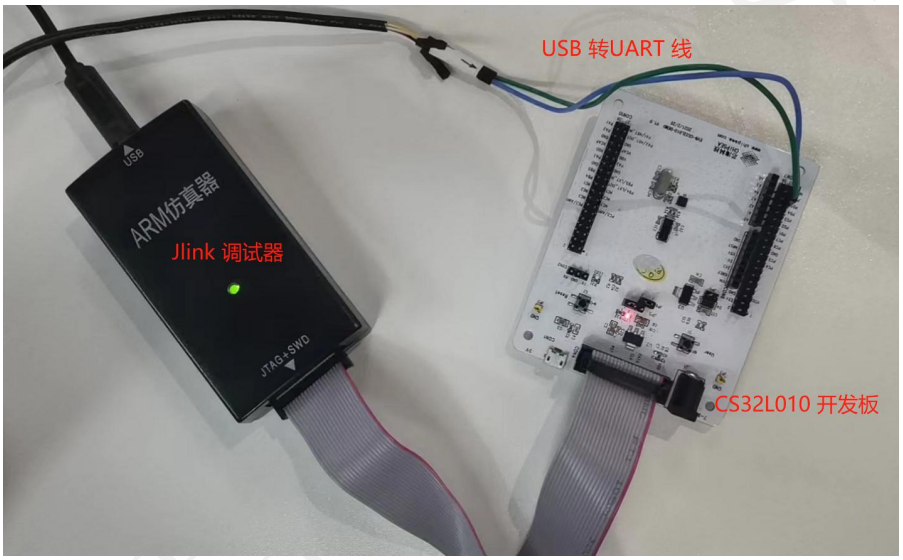


图 2.1 硬件工具

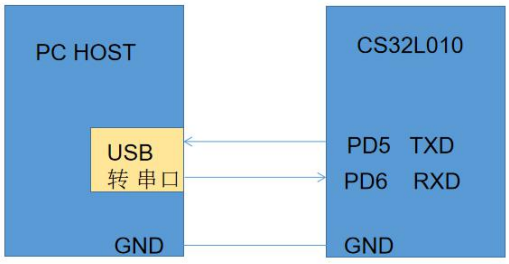


图 2.2 硬件连接说明

3 代码介绍

提供三个文件夹
APP1.1 是用户程序工程文件。
Tool 是 IAP 升级 PC 上位机工具。
UART_IAP 是 Boot 程序工程文件。

图 3.1 CS32L010 IAP 例程文件夹

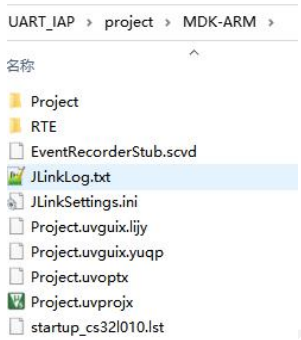


图 3.1 Boot App 工程文件路径

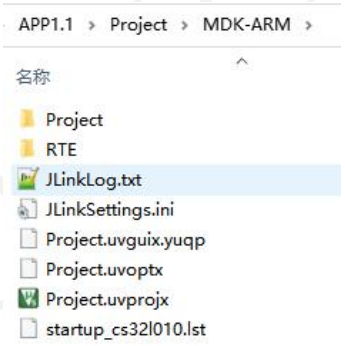


图 3.27 User App 工程文件路径

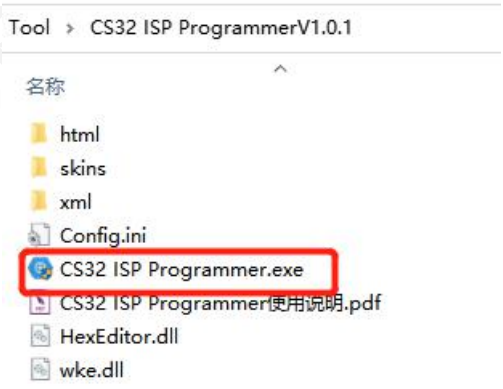


图 3.3 IAP 上位机

UART_IAP 工程文件代码介绍。

Main 函数中，首先初始化 硬件外设，然后初始化串口。 如果 MCU Flash 中已经有了用户程序，则调用延时函数，超时就直接跳到用户程序执行。如果没有用户程序，或者用户程序不完整，则执行用户程序升级。 如果超时或者接收到跳转命令，则直接跳到 用户程序执行。

```
int main(void)
{

	/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
	 HAL_Init();

	 /* Configure the system clock to HIRC 24MHz*/
	SystemClock_Config();
	 /* Configure uart1 for printf */
	LogInit();

	/* Enable UART INTERRUPT*/
	HAL_NVIC_EnableIRQ(UART1_IRQn);


	hal_timeout_disable();
	/*  App validate: Boot forced or invalid app - enter boot flow */
	if(csboot_app_validate() != false)
	{
		/* Timeout boot: Enable timeout */
		hal_timeout_enable();
	}
	while (1)
	{
		/* Polls and process command */
		ret = comm_process();

		/* Jump to app if timeout or received jump command */
		if((ret == RET_JUMP_TO_APP) || (hal_is_timeout() == 1))
		{
			csboot_app_jump_to_app();
		}

		/* Disable timeout if received validated command */
		if(ret == RET_OK)
		{
			hal_timeout_disable();
		}

	}
}
```

以下是串口初始化函数， PD5 配置成 TXD，PD6 配置成 RXD，波特率设置为 115200。

```
 #define LOG_SERIAL_BPS 115200


static void SerialInit(uint32_t baud_rate)
{
  huart1.Instance = UART1;
  huart1.Init.BaudRate = baud_rate;
  huart1.Init.BaudDouble = UART_BAUDDOUBLE_ENABLE;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;

  if(huart1.gState == HAL_UART_STATE_RESET)
  {
    /* Allocate lock resource and initialize it */
    huart1.Lock = HAL_UNLOCKED;

      GPIO_InitTypeDef GPIO_InitStruct = {0};
    /* Peripheral clock enable */
    __HAL_RCC_UART1_CLK_ENABLE();

      __HAL_RCC_GPIOD_CLK_ENABLE();
    /**UART1 GPIO Configuration
    PD5     ------> UART1_TXD
    PD6     ------> UART1_RXD
    */
```

```
    GPIO_InitStruct.Pin = GPIO_PIN_5;
    GPIO_InitStruct.Mode = GPIO_MODE_AF;
    GPIO_InitStruct.OpenDrain = GPIO_PUSHPULL;
    GPIO_InitStruct.Debounce.Enable = GPIO_DEBOUNCE_DISABLE;
    GPIO_InitStruct.SlewRate = GPIO_SLEW_RATE_HIGH;
    GPIO_InitStruct.DrvStrength = GPIO_DRV_STRENGTH_HIGH;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Alternate = GPIO_AF5_UART1_TXD;
    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = GPIO_PIN_6;
    GPIO_InitStruct.Mode = GPIO_MODE_AF;
    GPIO_InitStruct.Alternate = GPIO_AF5_UART1_RXD;
    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
  }

  huart1.gState = HAL_UART_STATE_BUSY;

  /* Set the UART Communication parameters */
  Log_UART_SetConfig(&huart1);

  /* Initialize the UART state */
  huart1.ErrorCode = HAL_UART_ERROR_NONE;
  huart1.gState= HAL_UART_STATE_READY;
  huart1.RxState= HAL_UART_STATE_READY;

}
```

以下跳转函数， UserApp 程序地址配置成 0x00002000 。

```
 #define CSBOOT_APP_BASE                                                    ((uint32_t)0x00002000)

void csboot_app_jump_to_app(void)
{

  uint32_t IapSpInitVal;

  IapSpInitVal = *(uint32_t *)CSBOOT_APP_BASE;
  IapJumpAddr = *(uint32_t *)(CSBOOT_APP_BASE + 4);

  if(MEM_ADDR_IN_RAM(IapSpInitVal) /*&& MEM_ADDR_IN_RANGE(IapJumpAddr)*/)//检查栈顶地址是否合法.
  {
    HAL_NVIC_DisableIRQ(UART1_IRQn);
    hal_timeout_disable();                    //<! Disable timeout
    __HAL_RCC_GPIOA_CLK_DISABLE();
    __HAL_RCC_GPIOB_CLK_DISABLE();
    __HAL_RCC_GPIOC_CLK_DISABLE();
    __HAL_RCC_GPIOD_CLK_DISABLE();
    __HAL_RCC_UART1_CLK_DISABLE();
//    SYSCFG->RMAPCFG |= SYSCFG_MEM_REMAP_SRAM;
    __set_MSP (IapSpInitVal);
    pIapFun = (void (*)(void))IapJumpAddr;
    (*pIapFun) ();
  }
}
```

APP1.1 （UserApp） 用户程序 Keil 工程文件配置.

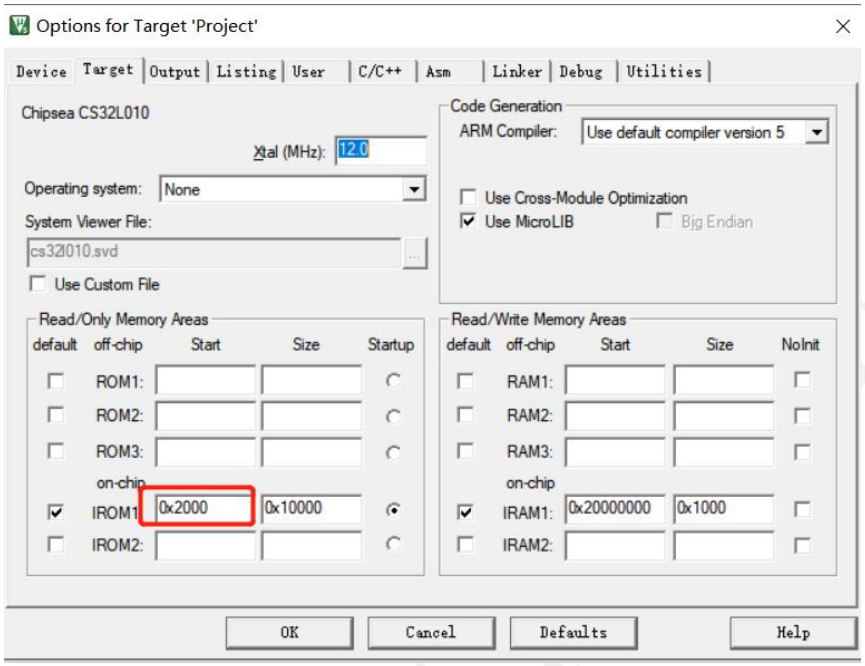


图 用户程序 KEIL 工程配置说明

用户程序 Mian 函数， PC3 电平翻转， PC3 在开发板上连接到一个 LED，并通过串口 PD5 打印信息出来。

设置#define VECT_TAB_OFFSET 0x00002000U /*!< Vector Table base offset field. */

```
/*!< Uncomment the following line if you need to relocate your vector Table in     Internal SRAM. */
/* #define VECT_TAB_SRAM */
//#define VECT_TAB_OFFSET  0x00000000U /*!< Vector Table base offset field. */
#define VECT_TAB_OFFSET  0x00002000U /*!< Vector Table base offset field. */
// IAP 功能，把此处地址修改，当切换正常程序时，影响到内部的 HAL_Delay()
//#define VECT_TAB_OFFSET  0x00002000U /*!< Vector Table base offset field. */
```

设置 LED GPIO

```
#define LED1_PIN                        GPIO_PIN_3
#define LED1_GPIO_PORT                  GPIOC

#define LED1_GPIO_CLK_ENABLE()                __HAL_RCC_GPIOD_CLK_ENABLE()
#define LED1_GPIO_CLK_DISABLE()                 HAL_RCC_GPIOD_CLK_DISABLE()
```

```
int main(void)
{
        /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
        HAL_Init();

        /* Configure the system clock to HIRC 24MHz*/
        SystemClock_Config();

        /* Initialize BSP Led for LED1 */
        __HAL_RCC_GPIOC_CLK_ENABLE();
        BSP_LED_Init(LED1);
```

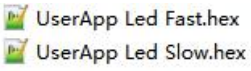
```
        //SysTick_Config(SystemCoreClock/10000); // 1KHZ

        /* Configure uart1 for printf */
        LogInit();
        printf("\nHello World\n");
        printf("SystemCoreClock = %d\n", SystemCoreClock);
        printf("Here is UserApp code @ 0x2000 for CS32L010 \n");

 while (1)
{
        // M0
        BSP_LED_Toggle(LED1);
        HAL_Delay(1000);
        printf("\nHello World, This is UserApp code  ++++\n");
}

        while (1);
}
```

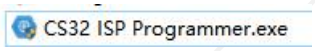
用户程序 Mian 函数， 修改 延时函数的参数，使得 LED 快速闪动和慢速闪动，并生成两个 hex 文件，用于后面做实测测试。



4 配合 PC IAP 上位机实际测试

首先打开 UART_IAP 文件下的 Boot 工程， 编译成功后下载到 开发板。

打开上位机升级工具 CS32 ISP Programmer.exe。
选择对应的串口，选择对应的芯片型号，选择要升级的用户程序 Hex 文档。



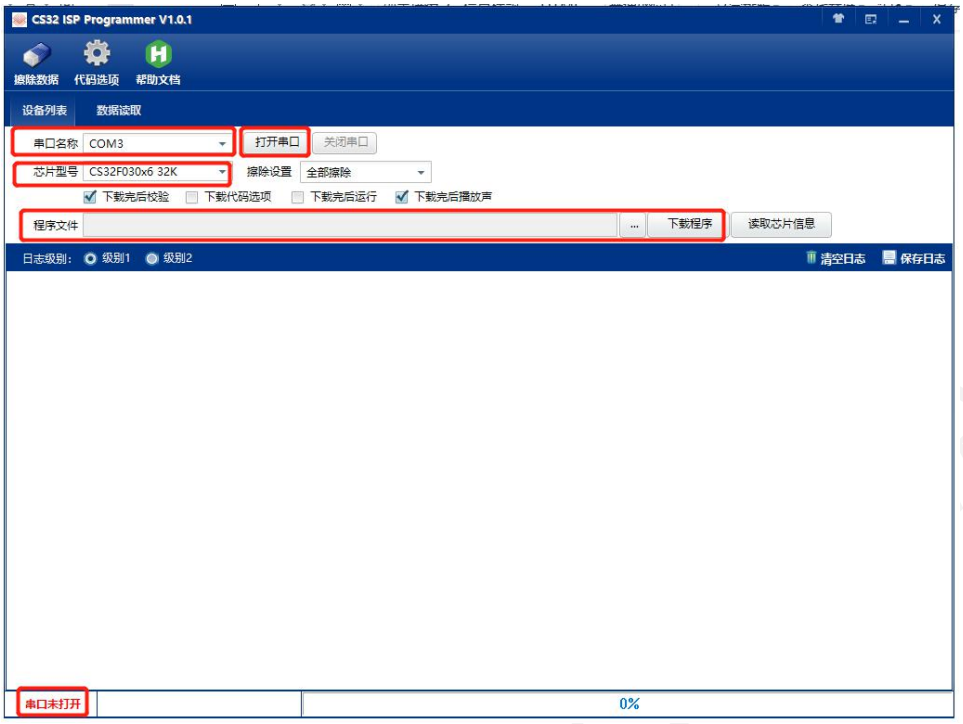


图 4.1 IAP 升级 PC 工具界面

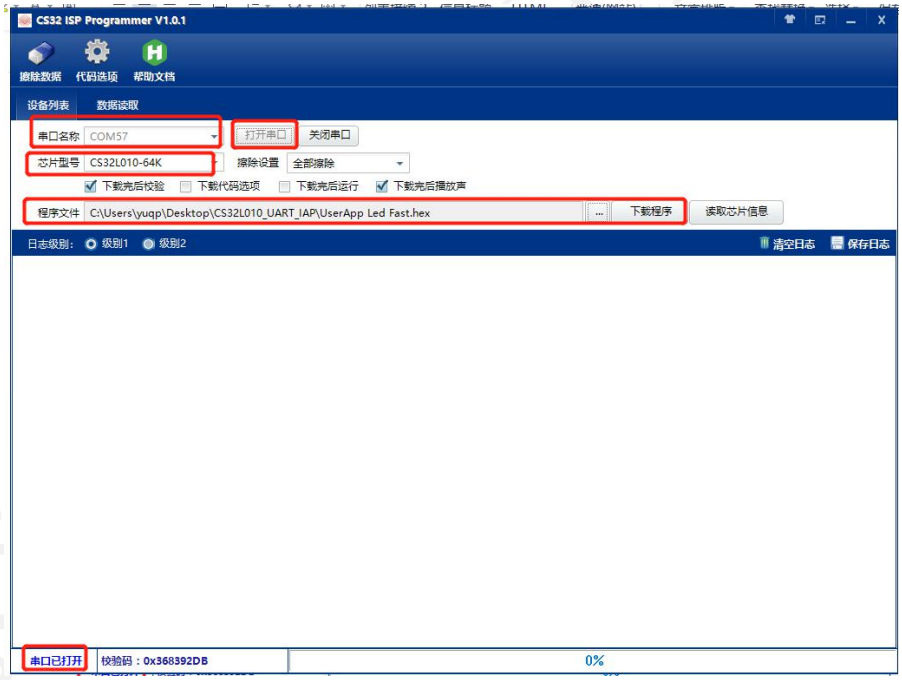


图 4.2 IAP 工具需要配置的参数

选择上一步生成 的 LED 闪灯 Hex 文档，选择下载数据。

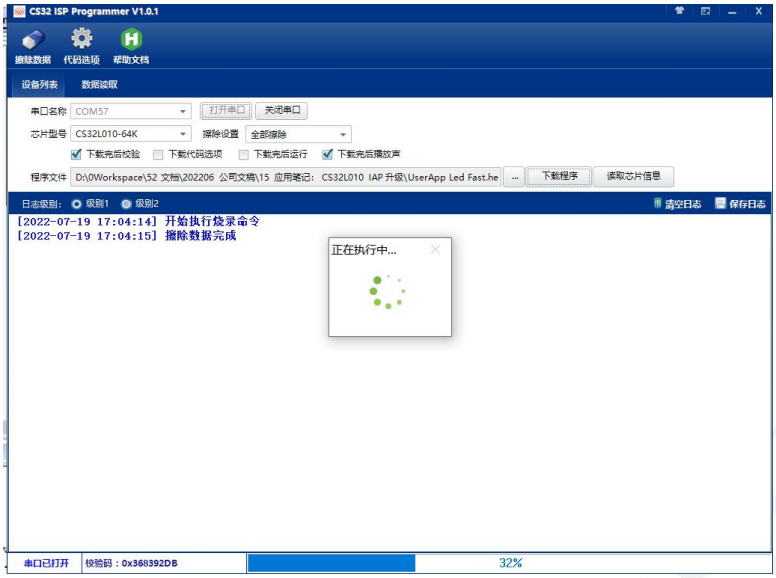


图 4.3 开始 IAP 升级程序

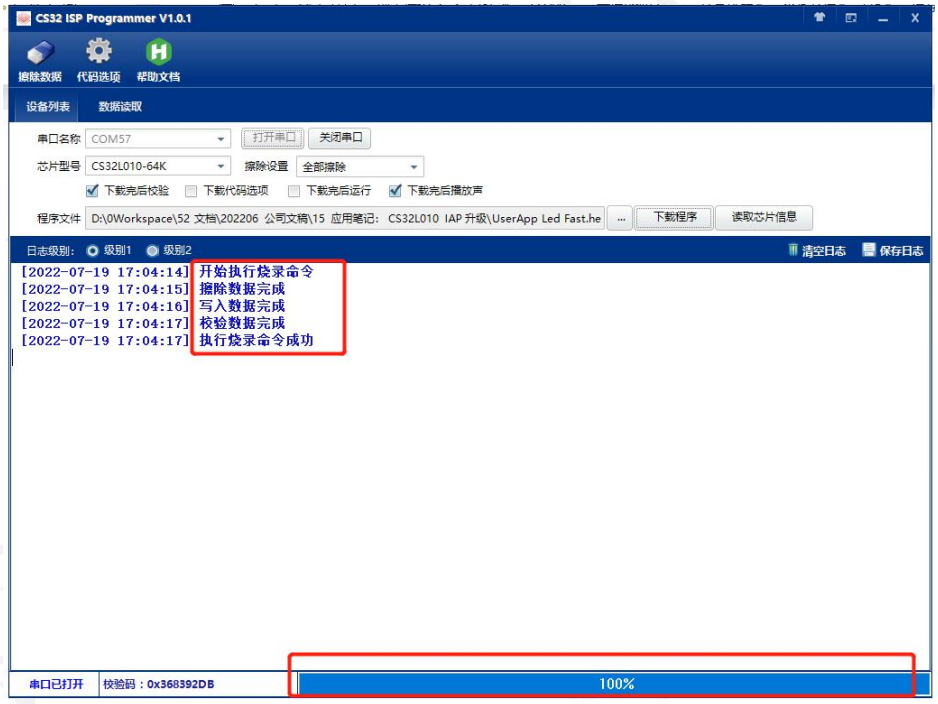


图 4.4 IAP 程序升级成功

IAP 程序升级成功， 按一下复位键，开发板上 LED 灯会闪动起来，同时串口会显示对应信息。

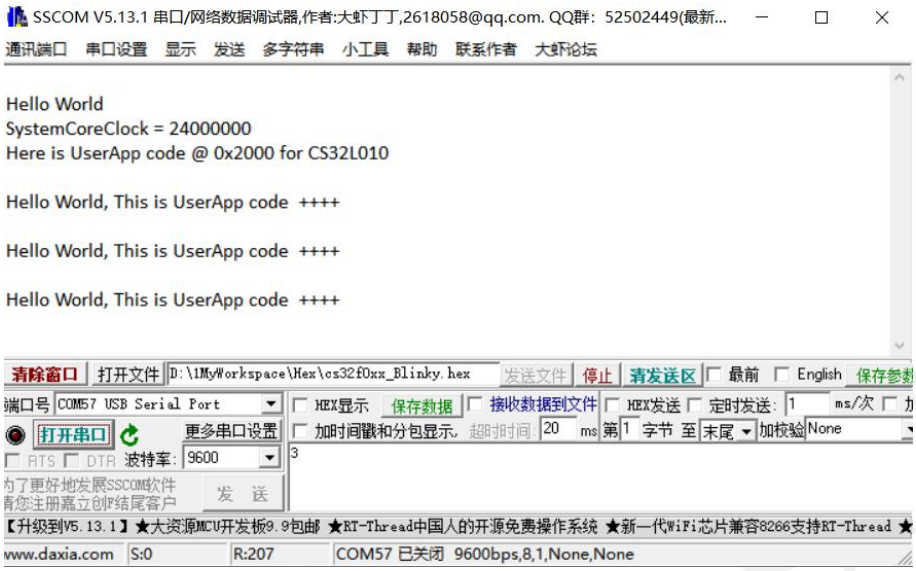


图 4.5 IAP 升级成功后， MCU 运行，打印串口信息

5 总结

本文档介绍了 CS32L010 MCU 通过 UART 接口来升级用户程序的基本功能，并且提供了参考的下位机 Bootloader 工程文件， 用户程序工程文件， 上位机 IAP 升级工程文件。用户可以根据自己的实际需要，增加相应功能来实现不同实际需求的应用。

上位机的工具 和下位机的参考代码，功能不是非常丰富，仅为需要开发 基于 UART IAP 升级程序的用户提供一下参考思路，方便用户评估 IAP 的功能。