# M032 ISP 更新 APROM 与 SPI Flash

NuMicro® 32 位系列微控制器范例代码介绍

## 文件信息

| | |
|---|---|
| 应用简述 | 本范例代码使用 ISP LDROM 程序更新 APROM 与 SPI Flash 中的数据 |
| BSP 版本 | M031_Series_BSP_CMSIS_V3.07.000 |
| 硬件平台 | NuMaker-M032KI V1.0 |

*The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

*For additional information or questions, please contact: Nuvoton Technology Corporation.*

www.nuvoton.com

# 1. 概述

APROM 通常储存用户代码，而 LDROM 则用于储存 Bootloader of In-System-Programming (ISP)。此 LDROM 范例代码通过 UART 接口接收用户代码，然后通过缓存控制将用户代码更新到 APROM 中，从而实现 In-System Programming (ISP) 功能。它还支持通过 SPI Flash 的指令集将资料更新到外部 SPI Flash。

## 1.1 原理

无论接收到的确认封包还是发送的命令封包，ISP 收发数据封包的大小均为 64 字节。数据封包通过 UART 接口发送和接收。

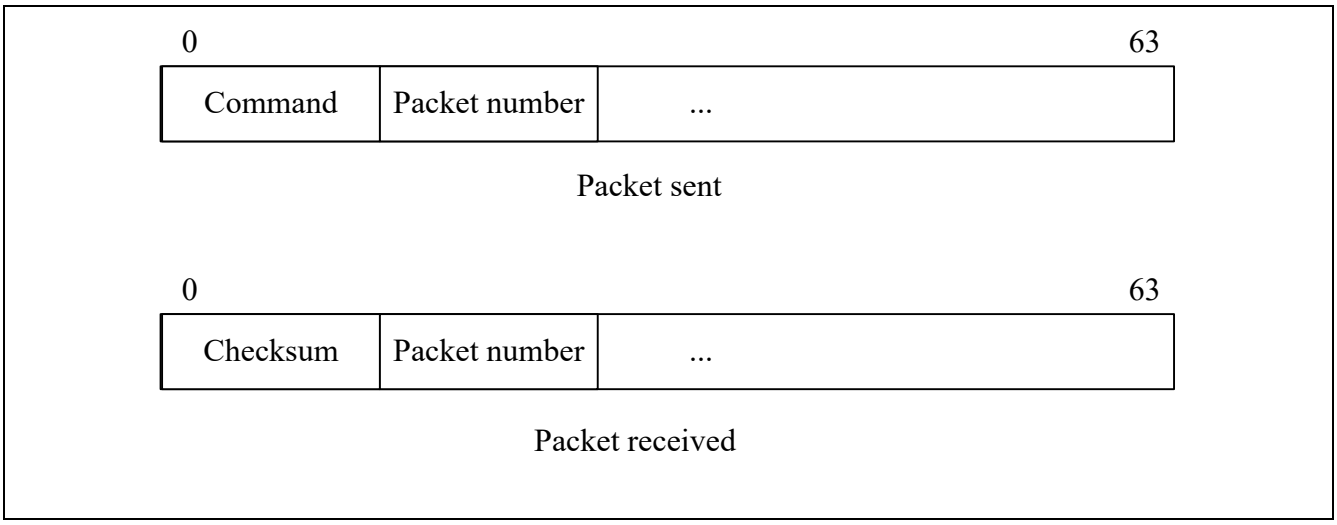


图 1-1 ISP 封包的发送和接收

### 1.1.1 ISP 命令集

ISP 支持的命令如下所示。

| Command | Command Code | ACK | Command Description |
|---|---|---|---|
| CMD_ERASE_SPI_FLASH | 0x000000D0 | Yes | 从偏移地址（默认为 0）开始擦除 64 KB SPI Flash，直到所有区块都被擦除。 |
| CMD_UPDATE_SPI_FLASH | 0x000000D1 | Yes | 从用户指定的地址开始对 SPI Flash 进行编程。 |

| | | | |
|---|---|---|---|
| CMD_GET_SPI_JEDEC | 0x000000D2 | Yes | 获取 SPI Flash 的 JEDEC ID |

表 1-1 ISP 命令集

**CMD_ERASE_SPI_FLASH 命令格式**

Host Send (unit by byte):

| 4 | 4 | 4 | 4 | n |
|---|---|---|---|---|
| CMD | Packet Number | Start Address | Data | Data…… |

Device ACK:

| 4 | 4 | |
|---|---|---|
| Checksum | Packet Number | Total Checksum |

此命令用于表示 ISP 对 SPI Flash 执行擦除操作。用户可以指定要擦除的起始地址。当 ISP 接收到命令封包后，会向 SPI Flash 发送区块擦除指令，以擦除 64 KB 的 SPI Flash 数据。在擦除操作完成后，ISP 会计算整个已接收封包的校验值，然后向主机发送 ACK 封包。

主机在接收到 ACK 封包后，会检查校验值是否正确；若校验值正确，则表示数据传输成功。主机将持续向 ISP 发送擦除命令，直到整颗 SPI Flash 全部擦除完成。

**CMD_UPDATE_SPI_FLASH 命令格式**

Host Send:

| 4 | 4 | 4 | 4 | n |
|---|---|---|---|---|
| CMD | Packet Number | Start Address | Length | Data…… |

Device ACK:

| 4 | 4 | |
|---|---|---|
| Checksum | Packet Number | Total Checksum |

此命令用于表示 ISP 对 SPI Flash 进行编程。用户可以指定要编程的起始地址。如果编程数据过大，无法在一个封包中完成传输，ISP Programming Tool 会将数据拆分为多个封包，通过多次传输完成。

ISP 会从接收到的封包中读取编程数据，并将其写入 SPI Flash 的指定地址。在编程操作完成后，ISP 会计算整个已接收数据封包的校验值，然后向主机发送 ACK 封包。

主机在接收到 ACK 封包后，会检查校验值是否正确；如果校验值正确，则表示数据传输成功。

**CMD_GET_SPI_JEDEC 命令格式**

Host Send:

| 4 | 4 |
|---|---|
| CMD | Packet Number |

Device ACK:

| 4 | 4 | 4 |
|---|---|---|
| Checksum | Packet Number | JEDEC ID |

此命令用于表示 ISP 读取 SPI Flash 的 JEDEC ID 信息，并将其传输给主机。

### 1.1.2 ISP 更新 SPI Flash 流程

更新 SPI Flash 的步骤如下：

1. 使用 Keil project 打开 **Chip Settings** 对话框，将 **Chips Booting Selection** 选项设置为 LDROM，然后点击 **OK**，如图 1-2 所示。

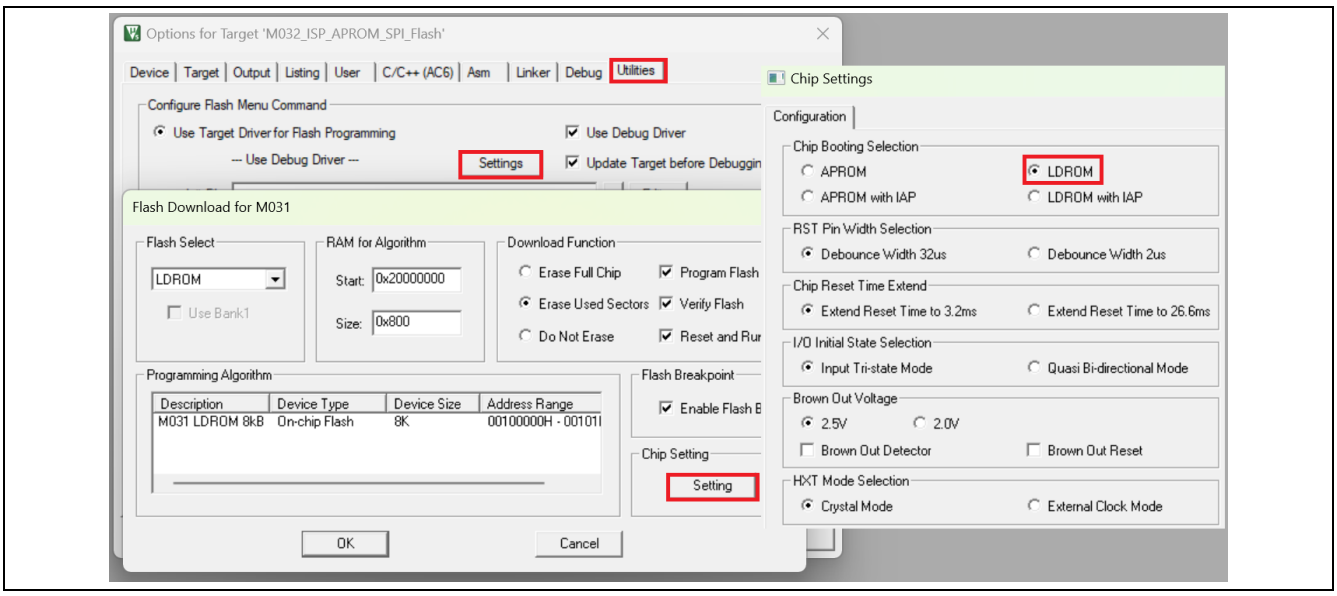


图 1-2 从 LDROM 启动的设定

2. 编译代码并将其下载到目标芯片的 LDROM 中。

3. 如图 1-3 所示，将 SPI Flash 的信息添加到 SPIFLASH ID.ini 文件中。例如 **EF4016** 是 Winbond W25Q32JV SPI Flash 的 JEDEC ID，**4M** 表示其储存容量为 4 MB。此步骤可确保 ISP Programming Tool 在连接到目标芯片时显示正确的 SPI Flash 容量。

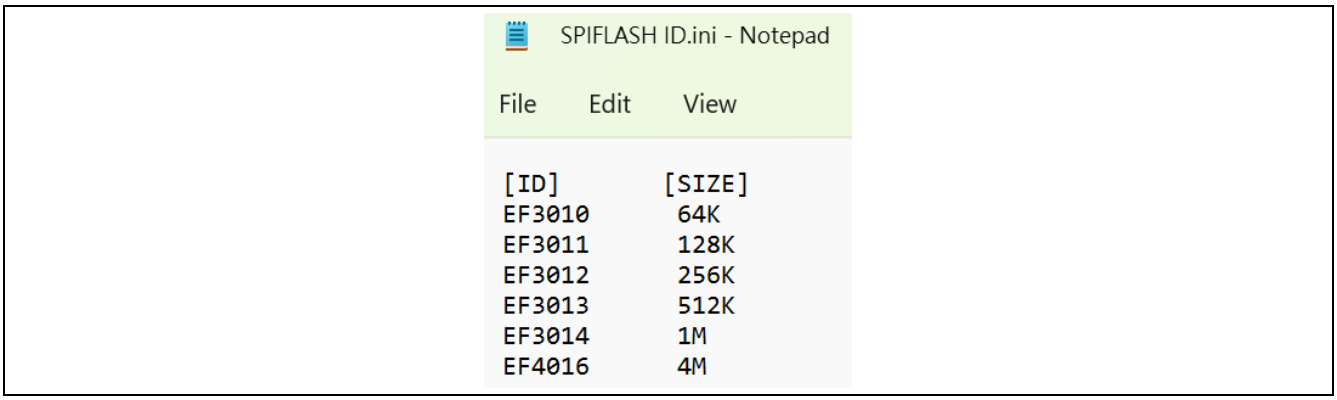


图 1-3 SPI Flash 信息文件

4. 打开 ISP Programming Tool，选择 UART 接口进行连接。

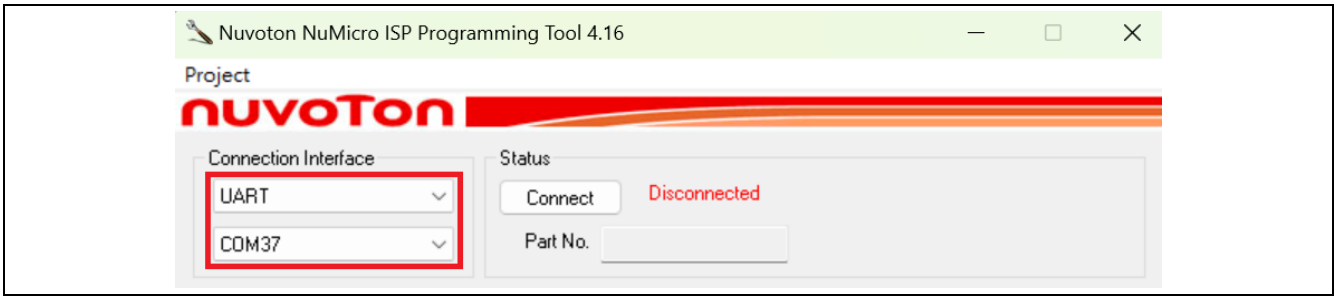


图 1-4 设置 ISP 工具的连接接口

5. 点击 **Project** 按钮，然后选择 **Open SPI Flash** 选项，如图 1-5 所示。

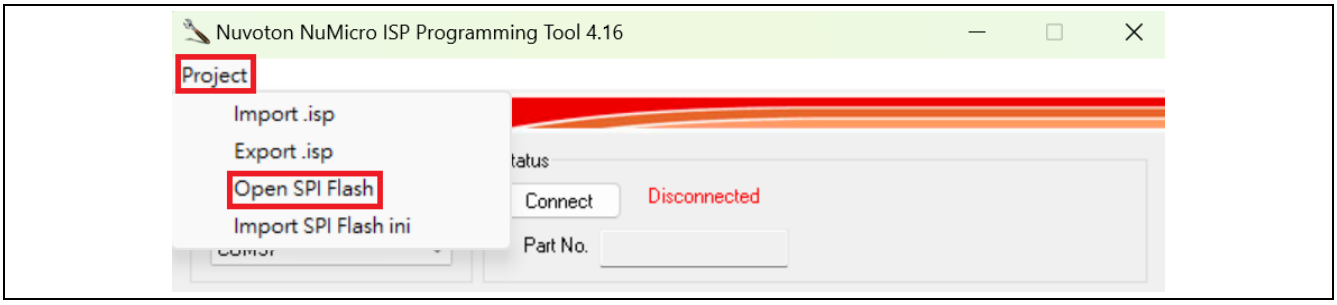


图 1-5 Open SPI Flash 选项

6. 点击 **Project** 按钮并导入 SPI Flash ini 文件，如图 1-6 所示。

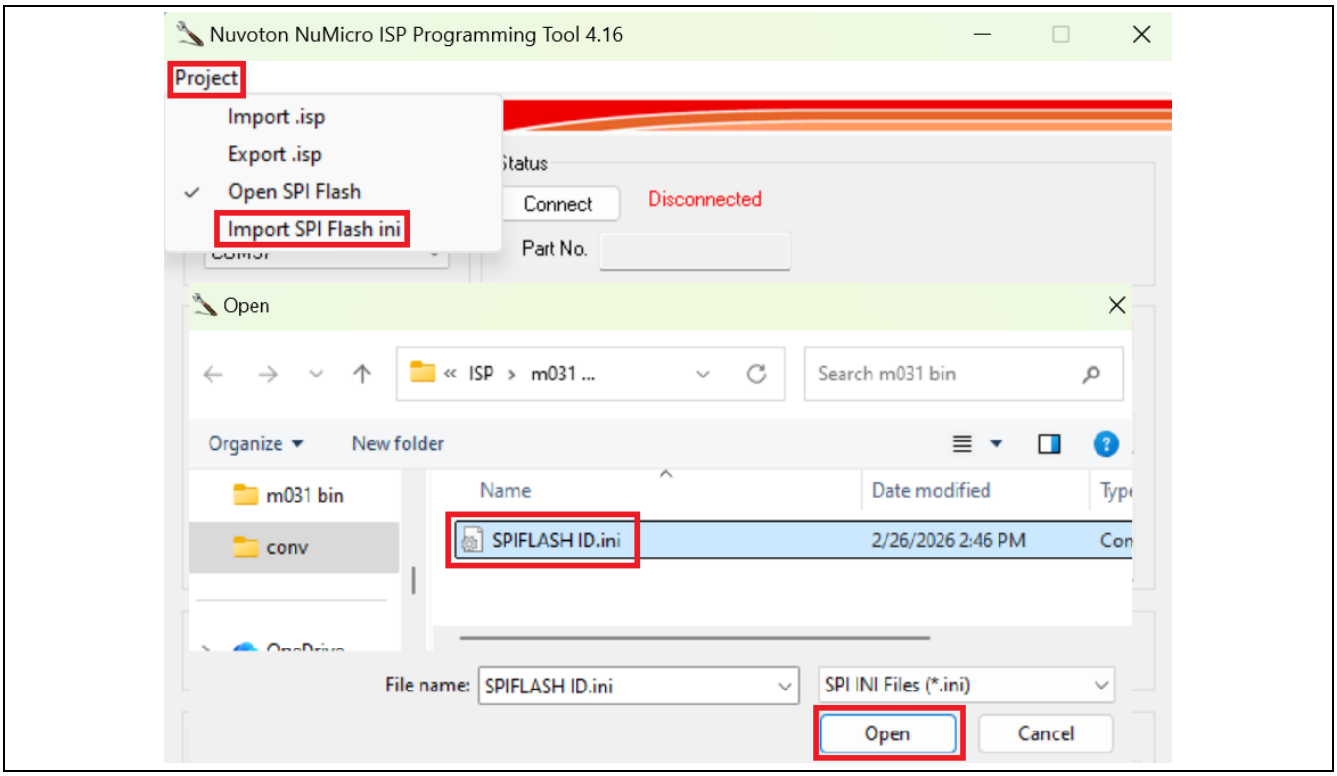


图 1-6 导入 SPI Flash 信息文件

7. 点击 **Connect** 按钮并重置目标芯片以运行 ISP 代码。ISP Programming Tool 将连接到目标芯片。

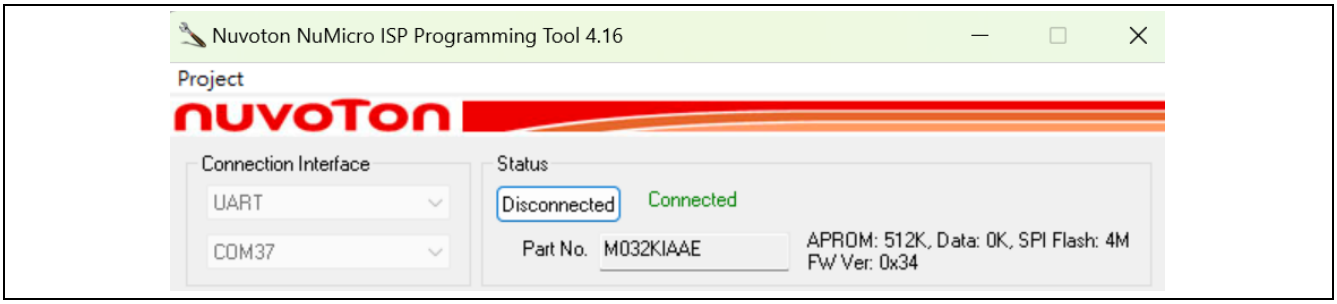


图 1-7 通过 UART 接口连接到目标芯片

8. 用户需要载入编程的文件，选择编程选项为 **SPI Flash**，然后点击 **Start** 按钮。ISP Programming Tool 将开始把文件写入到用户指定的位址，如图 1-8 所示。由于 ISP Programming Tool 在编程数据之前不会发出区块擦除命令，如果用户不确定 SPI Flash 的位址是否已被写过资料，请选择 **Erase SPI Flash** 选项，在编程前擦除整个 SPI Flash，以确保数据能够正确写入 SPI Flash。另请注意 **Base Address** 选项存在 2 KB 对齐限制。

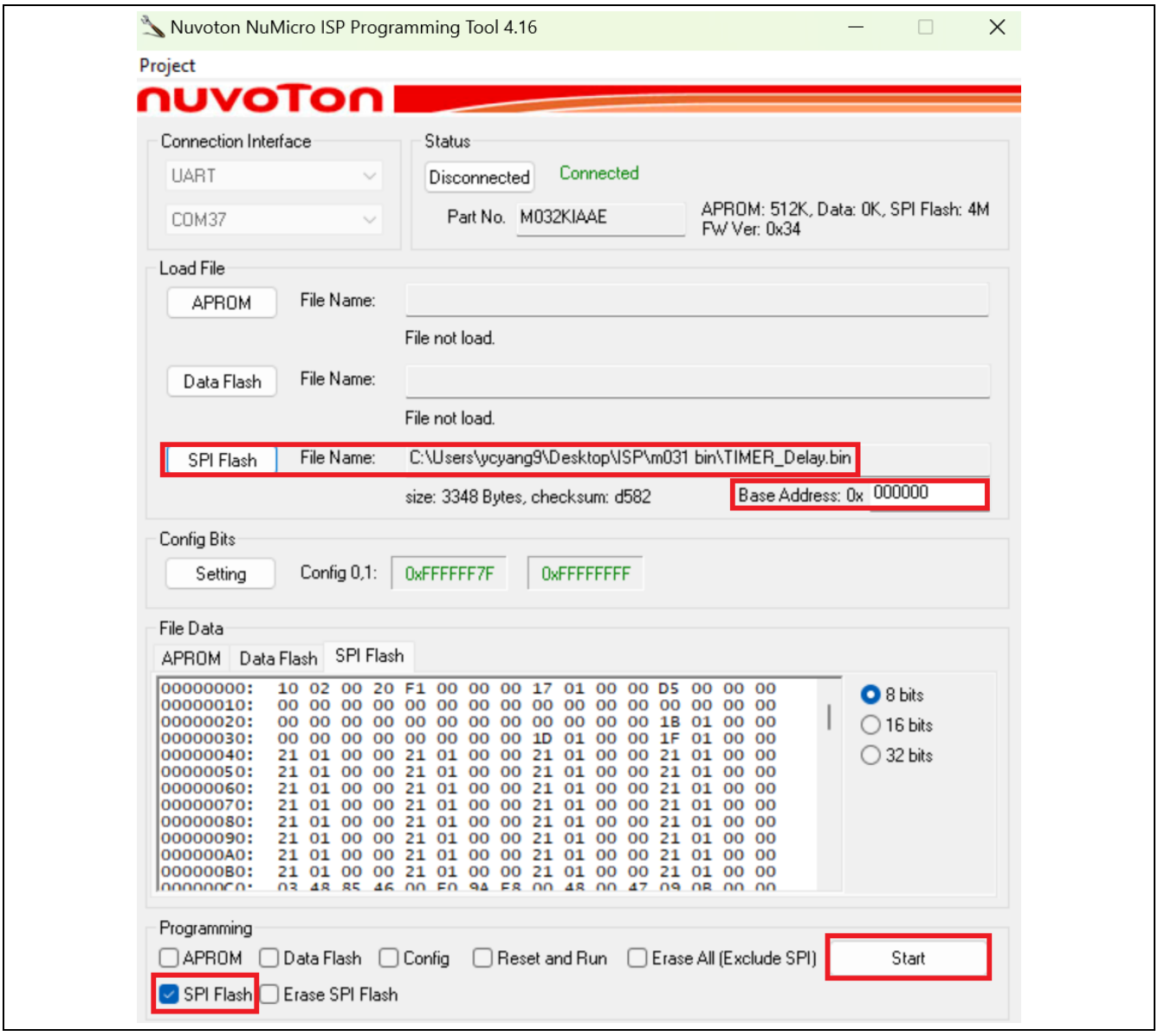


图 1-8 通过 UART 接口将数据写入外部 SPI Flash

## 1.2 执行结果

执行结果如下图所示。

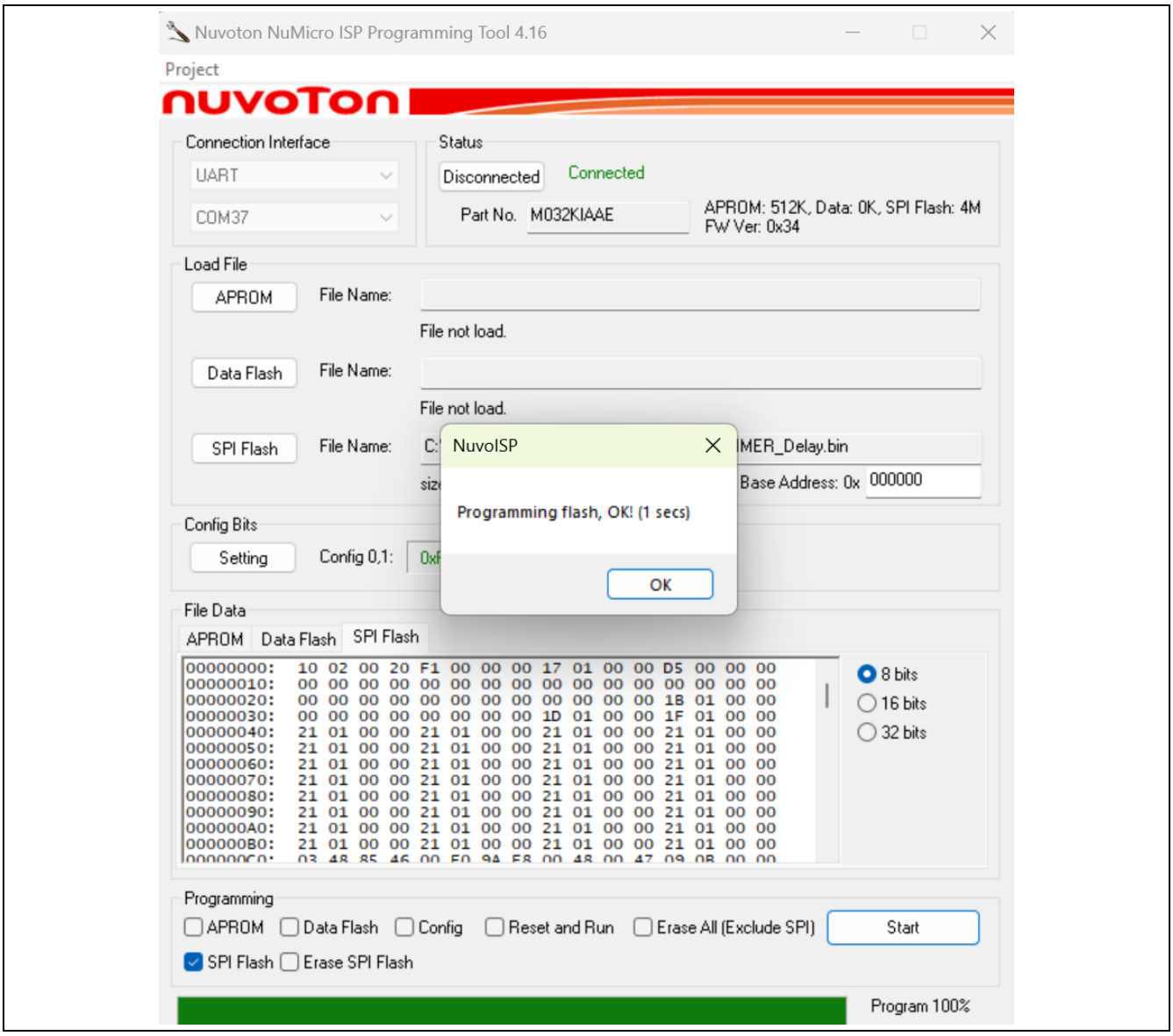
Nuvoton NuMicro ISP Programming Tool 4.16
Project
Connection Interface
UART
COM37
Status
Disconnected
Connected
Part No. M032KIAAE
APROM: 512K, Data: 0K, SPI Flash: 4M
FW Ver: 0x34
Load File
APROM
File Name:
File not load.
Data Flash
File Name:
File not load.
SPI Flash
File Name:
IMER_Delay.bin
Base Address: 0x 000000
NuvoISP
Programming flash, OK! (1 secs)
OK
Config Bits
Setting
Config 0,1:
File Data
APROM
Data Flash
SPI Flash
00000000: 10 02 00 20 F1 00 00 00 17 01 00 00 D5 00 00 00
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 1B 01 00 00
00000030: 00 00 00 00 00 00 00 00 1D 01 00 00 1F 01 00 00
00000040: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
00000050: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
00000060: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
00000070: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
00000080: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
00000090: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
000000A0: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
000000B0: 21 01 00 00 21 01 00 00 21 01 00 00 21 01 00 00
8 bits
16 bits
32 bits
Programming
APROM
Data Flash
Config
Reset and Run
Erase All (Exclude SPI)
Start
SPI Flash
Erase SPI Flash
Program 100%

图 1-9 烧录结果

## 2. 代码介绍

ISP Bootloader 的主程序设定系统频率并初始化 UART 和 SPI 功能，它将使用 UART 与 ISP Programming Tool 通信以进行更新。如果用户不需要更新 APROM 或 SPI Flash，则代码会将芯片设置为从 APROM 启动并执行系统复位。

```
int32_t main(void)
{
    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Init UART to 115200-8n1 */
    UART_Init();

    /* Configure SPI_FLASH_PORT as a master, MSB first, 8-bit transaction, SPI Mode-0
    timing, clock is 2MHz */
    SPI_Open(SPI_FLASH_PORT, SPI_MASTER, SPI_MODE_0, 8, 2000000);

    /* Disable auto SS function, control SS signal manually. */
    SPI_DisableAutoSS(SPI_FLASH_PORT);

    /* Checking if flash page size matches with target chip's */
    if ((GET_CHIP_SERIES_NUM == CHIP_SERIES_NUM_I) || (GET_CHIP_SERIES_NUM ==
    CHIP_SERIES_NUM_G))
    {
        if (FMC_FLASH_PAGE_SIZE != 2048)
        {
            /* FMC_FLASH_PAGE_SIZE is different from target device */
            /* Please enable the compiler option PAGE_SIZE_2048 in fmc.h */
            while(SYS->PDID);
        }
    }
    else
    {
        if (FMC_FLASH_PAGE_SIZE != 512)
        {
            /* FMC_FLASH_PAGE_SIZE is different from target device */
            /* Please disable the compiler option PAGE_SIZE_2048 in fmc.h */
            while(SYS->PDID);
        }
    }

    CLK->AHBCLK |= CLK_AHBCLK_ISPCKEN_Msk;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Enable FMC ISP function. Before using FMC function, it should unlock system
    register first. */
    FMC->ISPCTL |= (FMC_ISPCTL_ISPEN_Msk | FMC_ISPCTL_APUEN_Msk);

    g_apromSize = GetApromSize();
```

```
    GetDataFlashInfo(&g_dataFlashAddr, &g_dataFlashSize);
    SysTick->LOAD = 300000 * CyclesPerUs;
    SysTick->VAL   = (0x00);
    SysTick->CTRL = SysTick->CTRL | SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk;

    while (1)
    {
        if ((bufhead >= 4) || (bUartDataReady == TRUE))
        {
            uint32_t lcmd;
            lcmd = inpw(uart_rcvbuf);

            if (lcmd == CMD_CONNECT)
            {
                break;
            }
            else
            {
                bUartDataReady = FALSE;
                bufhead = 0;
            }
        }

        if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk)
        {
            SYS->RSTSTS = (SYS_RSTSTS_PORF_Msk | SYS_RSTSTS_PINRF_Msk);
            FMC->ISPCTL &= ~(FMC_ISPCTL_ISPEN_Msk | FMC_ISPCTL_BS_Msk);
            SCB->AIRCR = (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ);

            /* Trap the CPU */
            while (1);
        }
    }

    while (1)
    {
        if (bUartDataReady == TRUE)
        {
            bUartDataReady = FALSE;
            ParseCmd(uart_rcvbuf, 64);
            PutString();
        }
    }
}
```

此函数解析来自 ISP Programming Tool 的命令，以完成更新流程。

```
int ParseCmd(unsigned char *buffer, uint8_t len)
{
    static uint32_t StartAddress, TotalLen, LastDataLen;
    uint32_t lcmd, srclen, i, regcnf0, security;
    unsigned char *pSrc;
    static uint32_t gcmd;
    uint32_t u32Addr, u32NTx, u32Id;

    g_pu8Response = response_buff;
```

```
    pSrc = buffer;
    srclen = len;
    lcmd = inpw(pSrc);
    outpw(g_pu8Response + 4, 0);
    pSrc += 8;
    srclen -= 8;
    ReadData(Config0, Config0 + 16, (uint32_t *)(g_pu8Response + 8)); /* Read config */
    regcnf0 = *(uint32_t *)(g_pu8Response + 8);
    security = regcnf0 & 0x2;

    if (lcmd == CMD_SYNC_PACKNO)
    {
        g_packno = inpw(pSrc);
    }

    if ((lcmd) && (lcmd != CMD_RESEND_PACKET))
    {
        gcmd = lcmd;
    }

    if (lcmd == CMD_GET_FWVER)
    {
        g_pu8Response[8] = FW_VERSION;
    }
    else if (lcmd == CMD_GET_DEVICEID)
    {
        outpw(g_pu8Response + 8, SYS->PDID);

        ExitParseCmd(buffer, len);
        return (0);
    }
    else if (lcmd == CMD_RUN_APROM || lcmd == CMD_RUN_LDROM || lcmd == CMD_RESET)
    {
        SYS->RSTSTS = 3; /* Clear bit */

        /* Set BS */
        if (lcmd == CMD_RUN_APROM)
        {
            i = (FMC->ISPCTL & 0xFFFFFFFC);
        }
        else if (lcmd == CMD_RUN_LDROM)
        {
            i = (FMC->ISPCTL & 0xFFFFFFFC);
            i |= 0x00000002;
        }
        else
        {
            i = (FMC->ISPCTL & 0xFFFFFFFE);/* ISP disable */
        }

        FMC->ISPCTL = i;
        SCB->AIRCR = (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ);

        /* Trap the CPU */
        while (1);
    }
```

```
    else if (lcmd == CMD_CONNECT)
    {
        g_packno = 1;

        ExitParseCmd(buffer, len);
        return (0);
    }
    else if ((lcmd == CMD_UPDATE_APROM) || (lcmd == CMD_ERASE_ALL))
    {
        EraseAP(FMC_APROM_BASE, (g_apromSize < g_dataFlashAddr) ? g_apromSize :
        g_dataFlashAddr); /* Erase APROM */

        if (lcmd == CMD_ERASE_ALL)
        {
            EraseAP(g_dataFlashAddr, g_dataFlashSize);
            *(uint32_t *)(g_pu8Response + 8) = regcnf0 | 0x02;
            UpdateConfig((uint32_t *)(g_pu8Response + 8), NULL);
        }

        bUpdateApromCmd = TRUE;
    }

    if ((lcmd == CMD_UPDATE_APROM) || (lcmd == CMD_UPDATE_DATAFLASH))
    {
        if (lcmd == CMD_UPDATE_DATAFLASH)
        {
            StartAddress = g_dataFlashAddr;

            if (g_dataFlashSize)   /*g_dataFlashAddr*/
            {
                EraseAP(g_dataFlashAddr, g_dataFlashSize);
            }
            else
            {
                ExitParseCmd(buffer, len);
                return (0);
            }
        }
        else
        {
            StartAddress = 0;
        }

        TotalLen = inpw(pSrc + 4);
        pSrc += 8;
        srclen -= 8;
    }
    else if (lcmd == CMD_UPDATE_CONFIG)
    {
        if ((security == 0) && (!bUpdateApromCmd))   /* Security lock */
        {
            ExitParseCmd(buffer, len);
            return (0);
        }

        UpdateConfig((uint32_t *)(pSrc), (uint32_t *)(g_pu8Response + 8));
```

```
        GetDataFlashInfo(&g_dataFlashAddr, &g_dataFlashSize);

        ExitParseCmd(buffer, len);
        return (0);
    }
    else if (lcmd == CMD_RESEND_PACKET)     /* For APROM&Data flash only */
    {
        uint32_t PageAddress;
        StartAddress -= LastDataLen;
        TotalLen += LastDataLen;
        PageAddress = StartAddress & (0x100000 - FMC_FLASH_PAGE_SIZE);

        if (PageAddress >= Config0)
        {
            ExitParseCmd(buffer, len);
            return (0);
        }

        ReadData(PageAddress, StartAddress, (uint32_t *)aprom_buf);
        FMC_Erase_User(PageAddress);
        WriteData(PageAddress, StartAddress, (uint32_t *)aprom_buf);

        if ((StartAddress % FMC_FLASH_PAGE_SIZE) >= (FMC_FLASH_PAGE_SIZE - LastDataLen))
        {
            FMC_Erase_User(PageAddress + FMC_FLASH_PAGE_SIZE);
        }

        ExitParseCmd(buffer, len);
        return (0);
    }
    else if (lcmd == CMD_GET_JEDECID)
    {
        u32Id = SpiFlash_ReadJEDECID();
        outpw(g_pu8Response + 8, u32Id);

        ExitParseCmd(buffer, len);
        return (0);
    }
    else if (lcmd == CMD_ERASE_SPIFLASH)
    {
        u32Addr = inpw(pSrc);
        SpiFlash_EraseBlock(u32Addr);
    }
    else if (lcmd == CMD_UPDATE_SPIFLASH)
    {
        u32Addr = inpw(pSrc);
        u32NTx = inpw(pSrc + 4);
        pSrc += 8;
        SpiFlash_NormalPageProgram(u32Addr, u32NTx, pSrc);
        memset(pSrc, 0, u32NTx);
        SpiFlash_NormalRead(u32Addr, u32NTx, pSrc);
    }

    if ((gcmd == CMD_UPDATE_APROM) || (gcmd == CMD_UPDATE_DATAFLASH))
    {
        if (TotalLen < srclen)
```

```
        {
            srclen = TotalLen;/* Prevent last package from over writing */
        }

        TotalLen -= srclen;
        WriteData(StartAddress, StartAddress + srclen, (uint32_t *)pSrc);
        memset(pSrc, 0, srclen);
        ReadData(StartAddress, StartAddress + srclen, (uint32_t *)pSrc);
        StartAddress += srclen;
        LastDataLen =  srclen;
    }

    ExitParseCmd(buffer, len);
    return (0);
}
```

ParseCmd() 函数的这一部分用于更新 SPI Flash，并将 JEDEC ID 回复 ISP Programming Tool。

```
    else if (lcmd == CMD_GET_JEDECID)
    {
        u32Id = SpiFlash_ReadJEDECID();
        outpw(g_pu8Response + 8, u32Id);

        ExitParseCmd(buffer, len);
        return (0);
    }
    else if (lcmd == CMD_ERASE_SPIFLASH)
    {
        u32Addr = inpw(pSrc);
        SpiFlash_EraseBlock(u32Addr);
    }
    else if (lcmd == CMD_UPDATE_SPIFLASH)
    {
        u32Addr = inpw(pSrc);
        u32NTx = inpw(pSrc + 4);
        pSrc += 8;
        SpiFlash_NormalPageProgram(u32Addr, u32NTx, pSrc);
        memset(pSrc, 0, u32NTx);
        SpiFlash_NormalRead(u32Addr, u32NTx, pSrc);
    }
```

ParseCmd() 函数的这一部分用于更新 APROM 和 Data Flash。

```
    if ((gcmd == CMD_UPDATE_APROM) || (gcmd == CMD_UPDATE_DATAFLASH))
    {
        if (TotalLen < srclen)
        {
            srclen = TotalLen;/* Prevent last package from over writing */
        }

        TotalLen -= srclen;
        WriteData(StartAddress, StartAddress + srclen, (uint32_t *)pSrc);
        memset(pSrc, 0, srclen);
        ReadData(StartAddress, StartAddress + srclen, (uint32_t *)pSrc);
        StartAddress += srclen;
```

```
        LastDataLen =  srclen;
    }
```

# 3. 软件与硬件需求

## 3.1 软件需求

- BSP 版本
  - M031_Series_BSP_CMSIS_V3.07.000
- IDE 版本
  - Keil uVision 5.38

## 3.2 硬件需求

- 电路组件
  - NuMaker-M032KI V1.0
  - Level1-Training
- 引腳連接
  - 将 Nu-Link2-Me 的 USB 接口连接到 PC 的 USB 端口，以显示范例代码的执行结果。

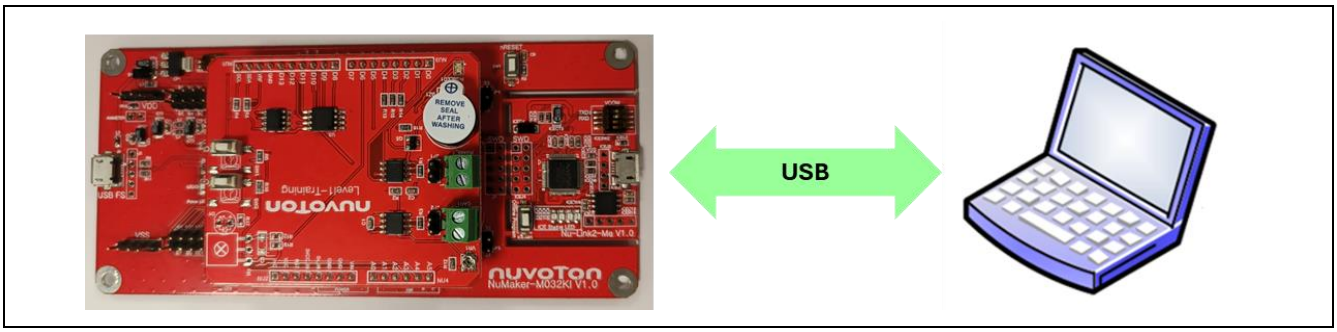


图 3-1 引腳連接

# 4. 目录信息

目录结构如下图所示。

| | |
|---|---|
| EC_M032_ISP_Update_APROM_and_SPI_Flash_V1.00 | |
| Library | Sample code header and source files |
| CMSIS | Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp. |
| Device | CMSIS compliant device header file |
| StdDriver | All peripheral driver header and source files |
| SampleCode | |
| ExampleCode | |
| *Project* | Source file of example code |
| WindowsTool | ISP Programming Tool |

图 4-1 目录信息

# 5. 范例程序执行

## 5.1 Keil MDK

1. 在 Windows 档案总管中进入 **Keil** 工程文件夹，再开启 *M032_ISP_Update_APROM_and_SPI_Flash*.uvprojx 工程文件。
2. 编译代码
3. 下载代码
4. 进入仿真模式界面
   - 执行代码

## 5.2 VS Code

1. 在 VS Code IDE 中开启 **File** 选单
   - 单击 **Open Folder**
   - 选取工程目录下的 **VSCode** 文件夹
2. 选择 **CMSIS**
   - 单击 **Build solution** 编译工程
   - 单击 **Load & Run application** 下载映像文件到目标芯片
   - 单击 **Load & Debug application** 进入仿真模式

# 6. 修订纪录

| Date | Revision | Description |
|---|---|---|
| 2026.06.23 | 1.00 | 初始发布。 |

## Important Notice

**Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".**

**Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.**

**All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.**

*Please note that all data and specifications are subject to change without notice.*
*All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*