M487 Secure ISP with AES and SPROM

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M487 to download encrypted firmware via the ISP Programming Tool, and then the M487 uses the AES crypto engine and SPROM to decrypt the firmware content and perform updates to the APROM, thereby completing a secure download process.
BSP Version	M480_BSP_CMSIS_V3.06.000
Hardware	NuMaker-PFM-M487 V3.0

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. Overview

This example code allows users to perform firmware updates via the UART interface. Users can first encrypt the firmware using Nuvoton CryptoTool , then use the Nuvoton NuMicro ISP Programming Tool to write the encrypted firmware to the NuMaker-PFM-M487. Upon successful reception, the M487 will invoke the decryption API within the SPROM, leveraging hardware encryption accelerators and a 256-bit AES key to decrypt the encrypted firmware. Subsequently, the firmware will be written to APROM to complete the entire update process.

1.1 Principle

The firmware encryption employs the Advanced Encryption Standard (AES), which is a symmetric encryption algorithm. It uses the same key for both encryption and decryption, with key lengths of 128, 192, or 256 bits. The data to be encrypted is divided into fixed-size blocks for encryption. Its advantages include high security and fast processing speed. AES is a U.S. National Institute of Standards and Technology (NIST) publication and is considered a national standard.

NIST offers various modes for the AES algorithm, and the Nuvoton CryptoTool utilizes AES-CFB-256 mode. Before performing file encryption or decryption, a 256-bit key and a 128-bit Initialization Vector (IV) need to be configured. Additionally, the location of the file to be encrypted, as well as the path and filename for the encrypted file need to be specified. Here's an example for reference:

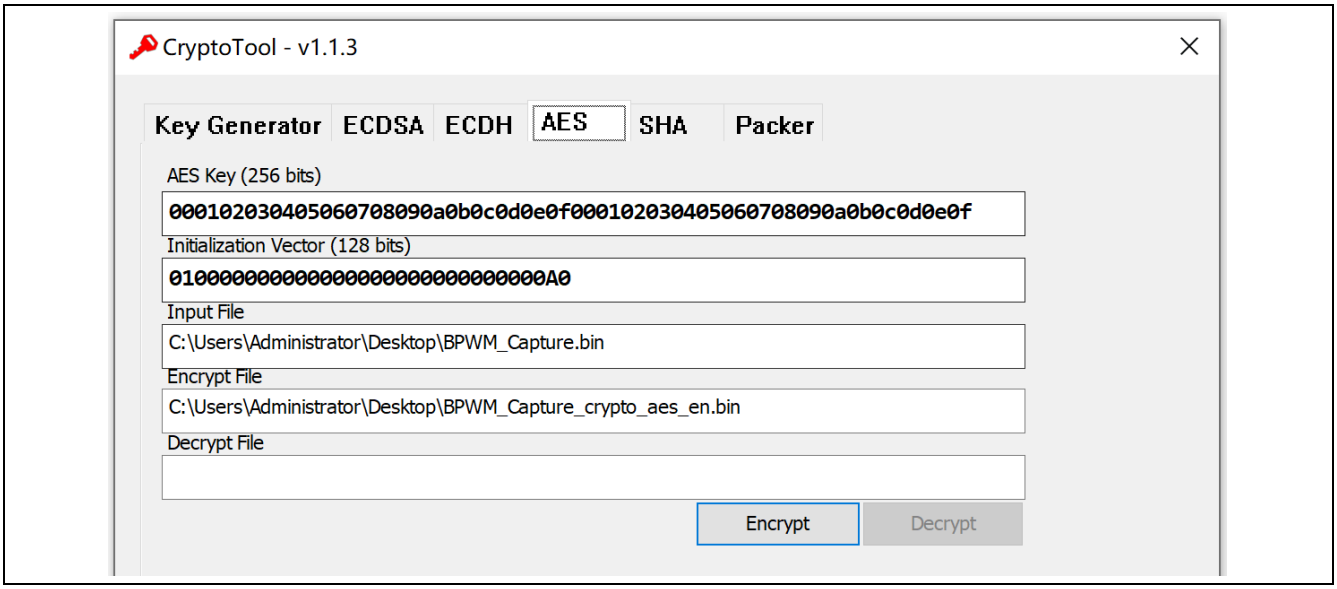


Figure 1-1 Nuvoton CryptoTool - v1.1.3

1.2 Security Protection Memory (SPROM)

The M487 can configure its internal SPROM as independent safety regions. The instructions (code) stored in these regions cannot be read but only executed, and up to 4 Kbytes for safety regions can be configured. To prevent accidental data access and leakage of sensitive software information, developers can position confidential code within these regions to protect it. For detailed information, please refer to section 6.5.4.4 of the M480 Series Technical Reference Manual. In the example code, the API used for decrypting firmware and the IV used for AES encryption/decryption will be placed within this area.

1.3 Demo Result

1.3.1 Program ISP Bootloader to MCU

The compilation results of this example code are shown in Figure 1-2, which shows to generate two bin files.

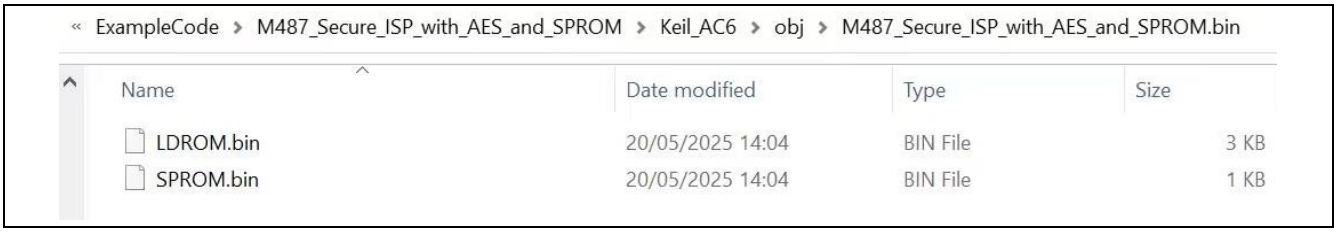


Figure 1-2 Compilation Results

As shown in Figure 1-3, use NuMicro ICP-Programing Tool for downloading the firmware. For LDROM, select “LDROM.bin”; for SPROM, select “SPROM.bin”.

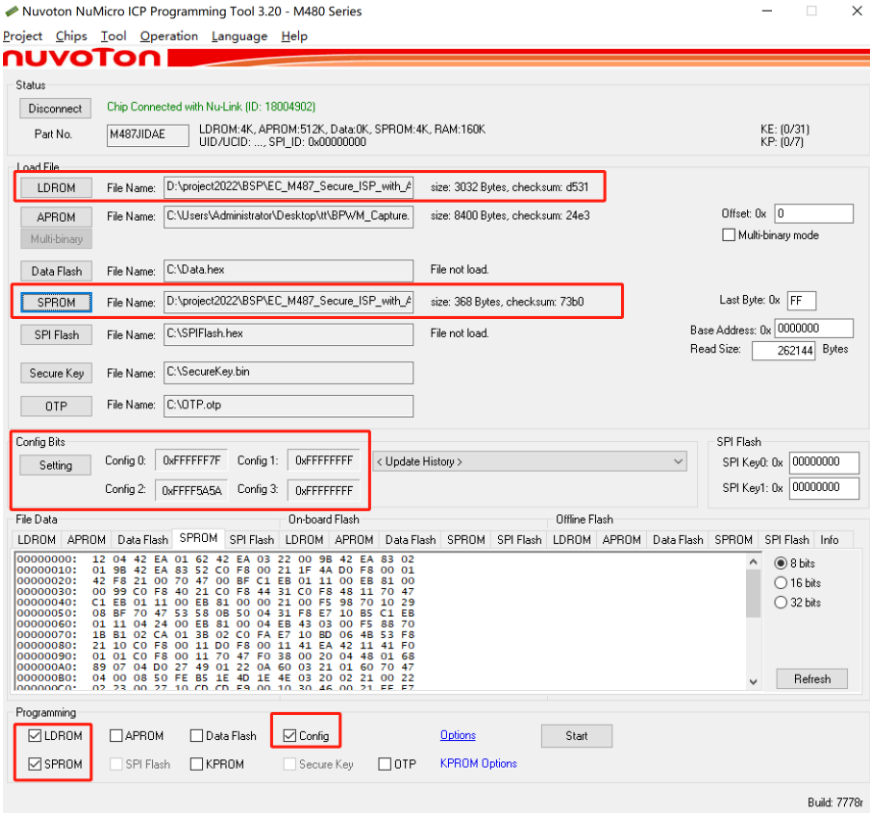


Figure 1-3 ICP Configuration

Chip settings are shown as follows.

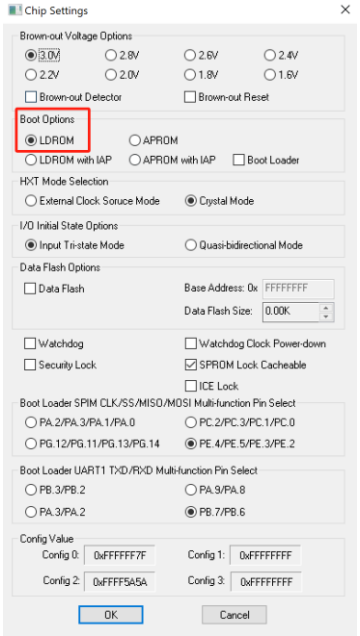


Figure 1-4 Chip Settings

1.3.2 Generate Encrypted Bin File

Then, use the Nuvoton CryptoTool to encrypt the file as shown in Figure 1-5. Here, select the AES field and configure AES Key as well as the Initialization Vector. Build any sample code in the M480 BSP (taking Template as an example), and then copy the compiled bin file to the CryptoTool folder. Set the paths for the Input File and Encrypt File, then click the **Encrypt** button. After a few seconds, you will see the encrypted bin file generated at that path.

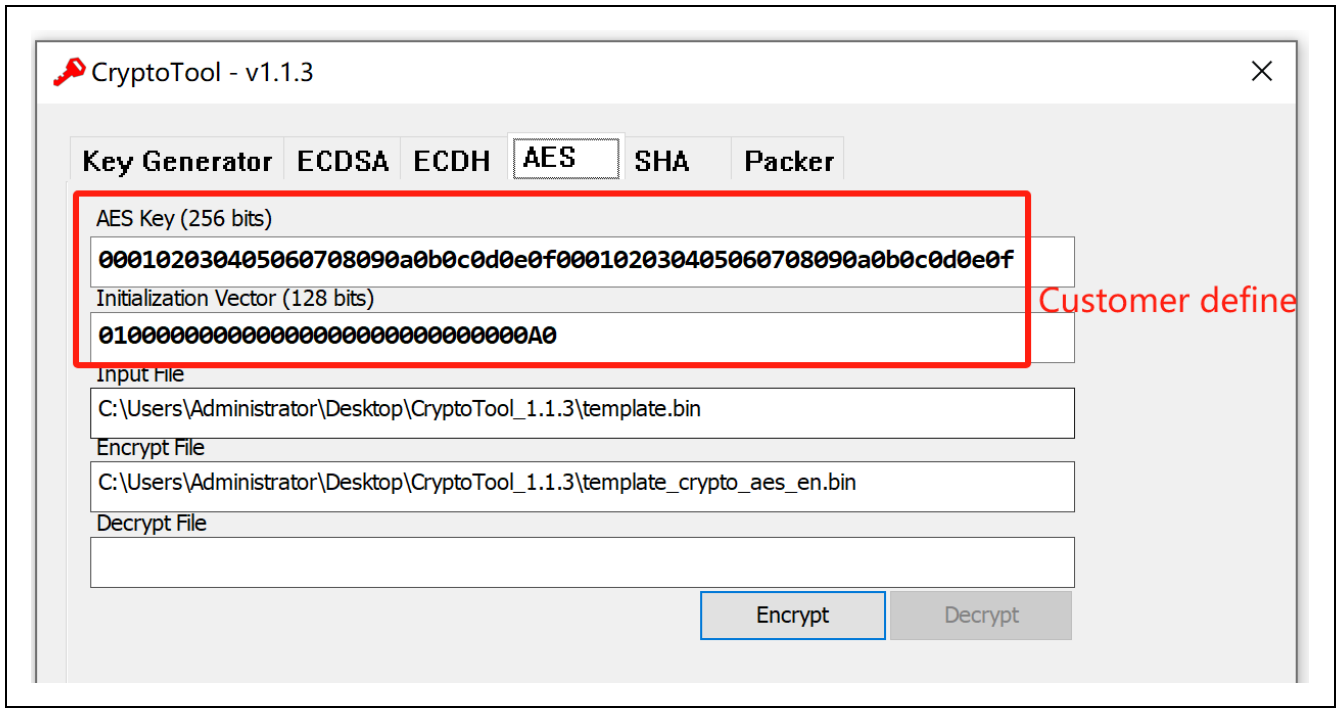


Figure 1-5 Encryption

1.3.3 Update APROM with ISP

Open the NuMicro ISP Programming Tool, and choose UART on the connection interface. Click **Connect** and then press the Reset button on the evaluation board to ensure connection with the tool. Click on APROM and load the encrypted update file "*template_crypto_aes_en.bin*". Choose the APROM setting in the Programming area and then click **Start**.

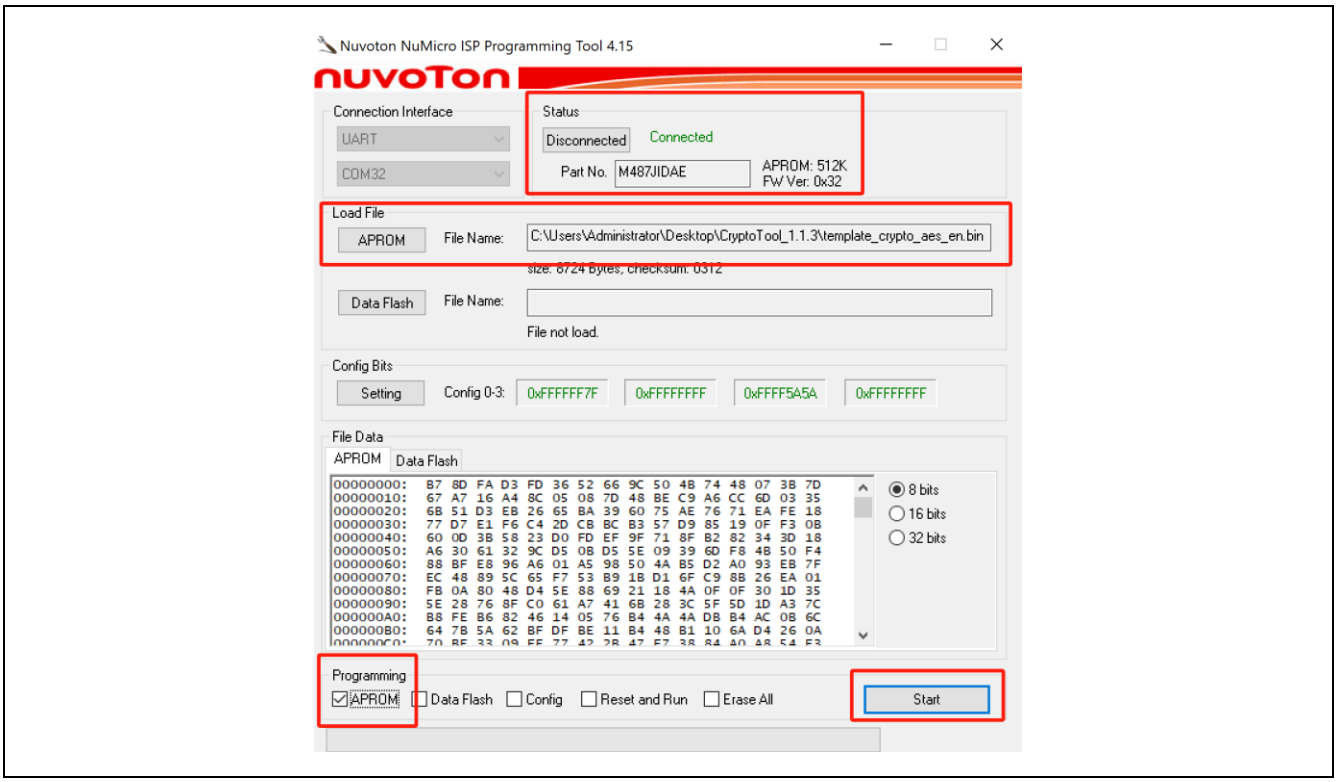


Figure 1-6 ISP Programming Tool

Close the NuMicro ISP Programming Tool, and open the UART tool. Then press the Reset button on the evaluation board. You will see the result "Hello World" shown in Figure 1-7 for the Template sample code in M480_BSP, which Indicates that the encrypted "*template_crypto-aes_en. bin*" file has been decrypted and programmed to APROM and running correctly.

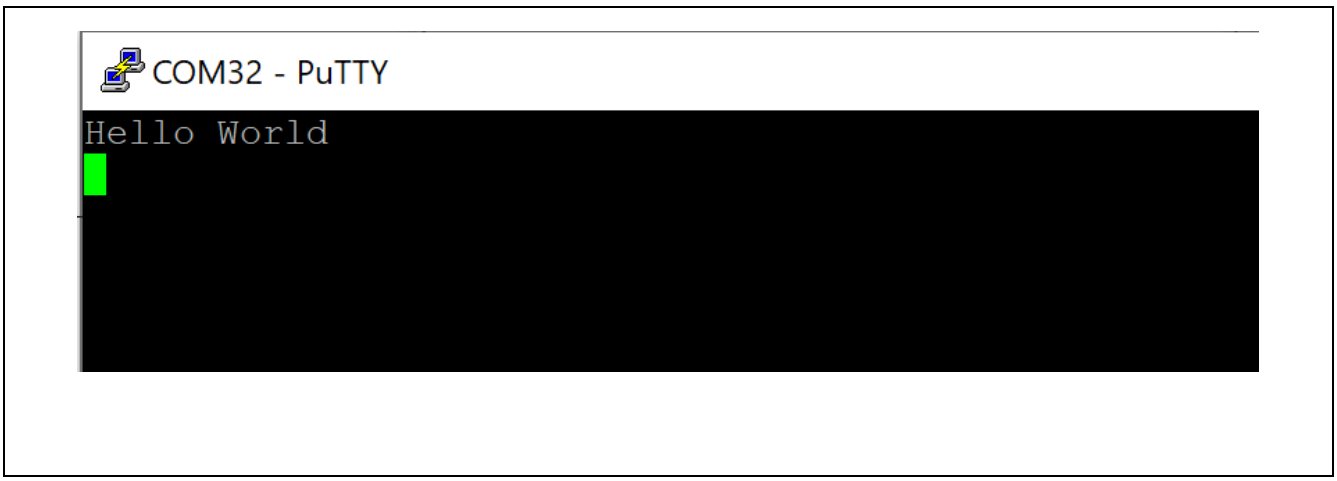


Figure 1-7 Template Program Execution Result

2. Code Description

The code main() below meets the architecture. The first half uses UART and ISP protocols to write encrypted firmware into the SRAM, while the second half implements the details of secure updates in void UpdateRoutine (void), protected by SPROM to prevent hackers from knowing the detailed process instructions. The code provides users with a basic example of security updates, and users can rewrite the Updating Routine process according to their specific needs, such as adjusting AES encryption and decryption modes and IV values, in order to successfully restore the original data. It should be noted that the SRAM area of M487 is only 160 Kbytes, so the APROM (bin file) size that can be updated in this example is limited.

```
int main(void)
{
    uint8_t u8SysTickCnt = 0;

    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();
    /* Init UART to 115200-8n1 */
    UART_Init();

    CLK->AHBCLK |= CLK_AHBCLK_ISPCKEN_Msk;
    FMC->ISPCTL |= FMC_ISPCTL_ISPEN_Msk;
    g_apromSize = GetApromSize();
    GetDataFlashInfo(&g_dataFlashAddr, &g_dataFlashSize);
     SYS_UnlockReg();                   /* Unlock protected registers */
    NVIC_EnableIRQ(CRPT_IRQn);
    AES_ENABLE_INT(CRPT);

    SysTick->LOAD = 0xFFFFFF;
    SysTick->VAL  = (0x00);
    SysTick->CTRL = SysTick->CTRL | SysTick_CTRL_CLKSOURCE_Msk
                    | SysTick_CTRL_ENABLE_Msk; /*using cpu clock*/

    g_u8ISPDone = FALSE;
    while (1)
    {

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

            if (lcmd == CMD_CONNECT)
            {
                goto _ISP;
            }
            else
            {
                bUartDataReady = FALSE;
                bufhead = 0;
            }
```

```
        }

        if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) /*timeout, then goto APROM*/
        {
            SysTick->VAL = 0;
            u8SysTickCnt++;
            if (u8SysTickCnt >= 5)
            {
                goto _APROM;
            }
        }
    }

_ISP:

    while (1)
    {
        if (bUartDataReady == TRUE)
        {
            bUartDataReady = FALSE;
            ParseCmd(uart_rcvbuf, 64);
            PutString();
        }
       if (g_u8ISPDone == TRUE)
        {
            UpdateRoutine();
            g_u8ISPDone = FALSE;
        }

    }

_APROM:
    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);
}
```

The code UpdateRoutine() below implements AES decryption files. With the core of implementing key writing and functional testing for AES engine encryption and decryption. The AES key is implemented through the au32MyAESKey array, and users can freely replace it with the key used by the actual product after knowing the operation. The AES key attribute is recommended to be set as a protected SPROM constant to ensure that the key can only be accessed by the AES engine, thereby achieving complete protection.

```
void UpdateRoutine(void)
{

    uint32_t StartAddress = 0x0000,  TotalLen;
    TotalLen = g_au32Encryptedimage[0];

    /*--------------------------------------
```

```
    *  step 1, decrypt image
    *  AES-256 CFB mode decrypt fw image
    *---------------------------------------*/
    AES_Open(CRPT, 0, 0, AES_MODE_CFB, AES_KEY_SIZE_256, AES_IN_OUT_SWAP);
    /* Use key in key store */
    AES_SetKey(CRPT, 0, (uint32_t *)au32MyAESKey, AES_KEY_SIZE_256);
    AES_SetInitVect(CRPT, 0, (uint32_t*)au32MyAESIV);
    AES_SetDMATransfer(CRPT, 0, (uint32_t)&g_au32Encryptedimage[1],
                       (uint32_t)au8OutputData, TotalLen);
    g_AES_done = 0;
    AES_Start(CRPT, 0, CRYPTO_DMA_ONE_SHOT);
    while (!g_AES_done);

    /* step 2, update APROM */
    FMC_ENABLE_AP_UPDATE();
    EraseAP(FMC_APROM_BASE,g_apromSize); /* erase APROM */
    WriteData(StartAddress, StartAddress + TotalLen, (uint32_t *)au8OutputData);

    return;
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480_BSP_CMSIS_V3.06.000
- IDE version
  - Keil uVersion 5.41

3.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M487 V3.0
- Pin Connect
  - Connect the UART0 pin to the PC UART to connect with the ISP Programming Tool and show the execution results of this example code.

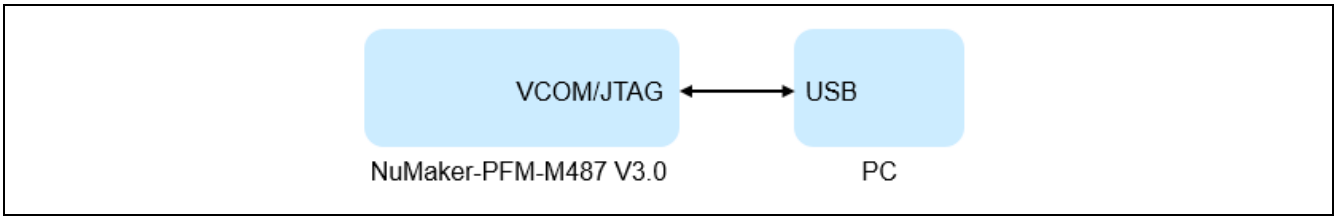


Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_ M487_Secure_ISP_with_AES_and_SPROM_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	Source file of example code

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *M487_Secure_ISP_with_AES_and_SPROM.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2025.04.24	1.00	Initial version.

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.