M2351 USB 组件固件更新

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	此范例程序展示了在 M2351 开发平台上实现组件固件更新 (CFU) 功能的方式。
BSP 版本	M2351_Series_BSP_CMSIS_V3.00.006
开发平台	NuMaker-PFM-M2351

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

此范例程序使用 NuMaker-PFM-M2351 开发平台，展示一个支持组件固件更新 (CFU) 的 USB 装置。当安装了来自 Windows Update 的更新信息的计算机连接到该 USB 装置时，CFU 将启动，开始更新该 USB 装置的子组件。

1.1 NuMaker-PFM-M2351 开发平台

NuMaker-PFM-M2351 是一款基于 NuMicro®-M2351 微控制器的开发板。Nuvoton NuMicro® M2351 安全微控制器系列采用 Arm® Cortex®-M23 核心，并搭载适用于 Armv8-M 架构的 TrustZone® 技术，可将传统的固件安全性提升至更高层次的强大软件安全性。

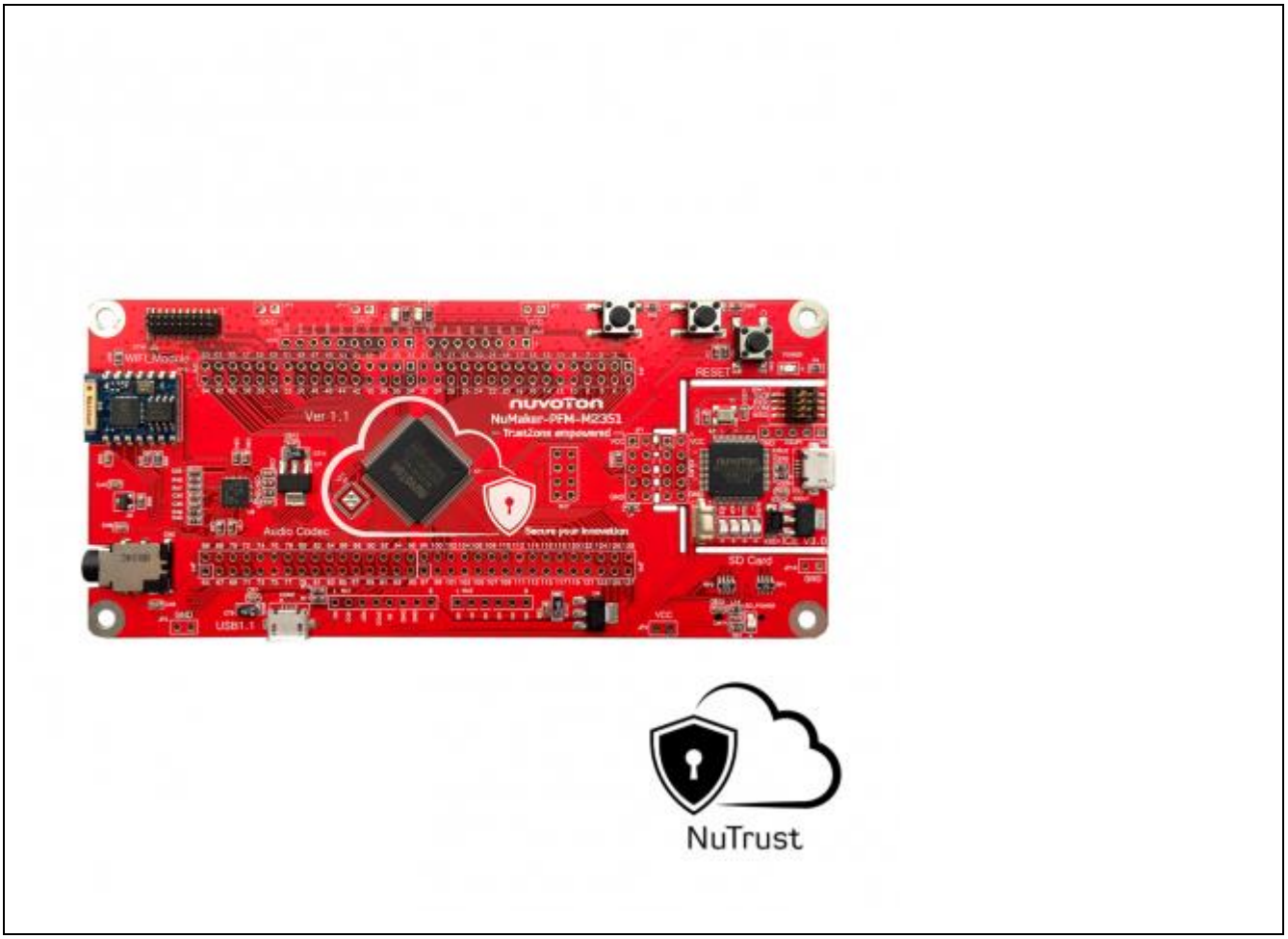


图 1-1 NuMaker-PFM-M2351 V1.0 开发平台

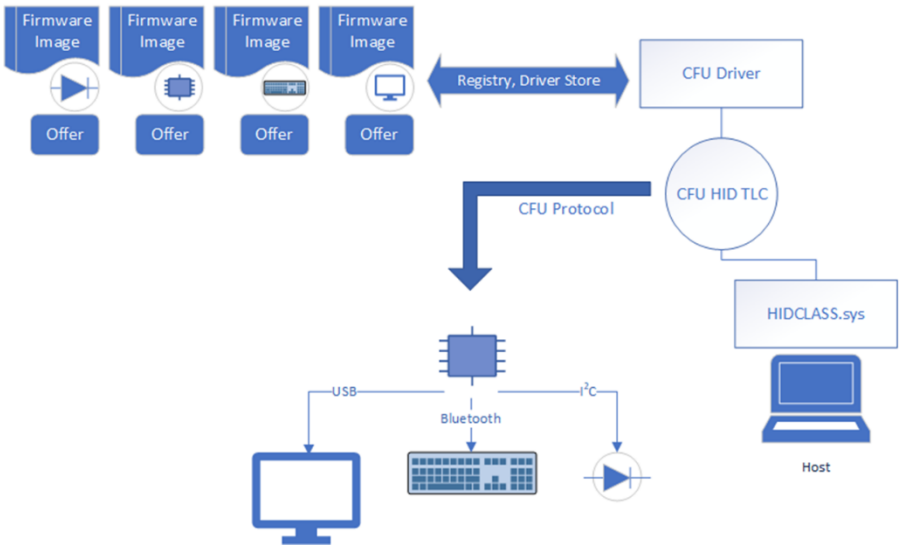
1.2 CFU 工作原理分析

1.2.1 背景介绍

组件固件更新 (CFU) 是 Microsoft 提供的一项服务，用于自动更新硬件固件，以提升装置的性能和安全性。CFU 的更新过程几乎不会打扰用户，甚至无需进入「更新模式」，用户无需等待或意识到更新正在进行。此外，CFU 基本上是自动更新：系统会自动检测并安装最新的固件更新，而无需用户干预。.

CFU 是通过 Windows Update 驱动程序提供的。而目标装置必须能够在下载不适合的固件之前将其拒绝。

1.2.2 CFU 进程开始前的示意图



https://blogs.windows.com/windowsdeveloper/2018/10/17/introducing-component-firmware-update/

图 1-2 架构示意图

1. 固件提供商通过 Windows Update 上传其驱动程序。
2. 客户端在主机端接收驱动程序。
3. 当与驱动程序设定相符的目标装置连接到主机时，主机将启动 CFU 进程。
4. 在 CFU 进程中，子组件的固件将通过接收来自主要目标组件的数据进行更新。

1.2.3 传输数据格式详情

CFU 提交给常驻固件的数据是一对档案，其中一个是「提案部分」，另一个是「内容部分」。CFU 驱动程序（主机）可能包含主要组件及其相关子组件的多个固件映像档。

提案部分包含足够的负载信息，以便主要组件决定该负载是否可接受。

提案信息包括 CFU 协议版本、组件 ID（如果适用，也包含子组件 ID）、固件版本、释出版本与仿真版本状态，以及其他相关信息。

封包的负载基本上是一组地址和固定大小的字节数组，用于进行编程操作。这些字节对主机而言是不可见的。

1.2.4 CFU 过程详情

使用 CFU 进行固件更新的一般流程是由主机向主要组件发送每个封包的提案。

当主机检测到与固件匹配的装置已连接时，主机会向装置发送「主机初始化通知」，告知装置 CFU 更新过程已准备就绪并等待接收。随后，主机会发送「提案开始通知」。

接着，主机将按顺序发送组件的提案。对于每个独立的组件提案，主要组件可以选择接受、拒绝或跳过该提案。如果提案被接受，主机将开始发送更新数据并启动更新过程；否则，主机将跳过并依次发送其他组件的提案。

这个过程将持续进行，直到装置对提案链中的每个组件提案都返回「拒绝」时为止。然后主机会发送「提案结束通知」，并完成 CFU 过程。

1.3 USB 描述符分析

1.3.1 组件 USB 描述符概述

USB 描述符是一组结构化数据，用于描述 USB 装置的属性、功能和需求。这些描述符帮助主机识别装置类型、可用功能以及传输需求。

1.3.2 描述符在组件装置中的作用

在处理 CFU 过程的组件装置中，描述符提供了装置如何与主机交互以及传输数据的详细信息。以下概述了每个描述符在视频传输中的具体作用及其在程序中的实现。

1.3.2.1 装置描述符

装置描述符提供有关 USB 装置的基本信息，例如装置类型、厂商 ID、产品 ID 以及支持的 USB 版本。这是主机获取装置信息的第一步，使主机能够识别装置的基本属性。

```
static uint8_t s_au8DeviceDescriptor [] = {
    LEN_DEVICE,     /* bLength */
    DESC_DEVICE,    /* bDescriptorType */
#ifdef SUPPORT_LPM
    0x01, 0x02,     /* bcdUSB >= 0x0201 to support LPM */
#else
    0x10, 0x01,     /* bcdUSB */
#endif
    0x00,           /* bDeviceClass */
    0x00,           /* bDeviceSubClass */
    0x00,           /* bDeviceProtocol */
    EP0_MAX_PKT_SIZE,   /* bMaxPacketSize0 */
    /* idVendor */
    USBD_CFU_VID & 0x00FF,
    (USBD_CFU_VID & 0xFF00) >> 8,
    /* idProduct */
    USBD_CFU_PID & 0x00FF,
    (USBD_CFU_PID & 0xFF00) >> 8,
    0x00, 0x00,     /* bcdDevice */
    0x01,           /* iManufacture */
    0x02,           /* iProduct */
    0x00,           /* iSerialNumber - no serial */
    0x01            /* bNumConfigurations */
};
```

1.3.2.2 配置描述符

配置描述符描述装置的配置选项，包括电源使用情况和支持的界面数量。主机利用此描述符来配置装置，以确保其正常运作。该描述符还包含 HID 描述符、界面描述符和端点描述符。

界面描述符 提供了关于配置中每个界面的详细信息，包括它们提供的功能（例如视频控制或视频流）以及使用的端点数量。

端点描述符 定义主机与装置之间数据传输的具体方法。对于视频流，端点通常配置为同步传输，以确保数据稳定性和实时传输。在范例程序代码中，使用了端点 2 和端点 3，因此它们的信息必须包含在端点描述符中。

程序代码实现：

```
cstatic uint8_t s_au8ConfigDescriptor[] =
{
    LEN_CONFIG,     /* bLength */
    DESC_CONFIG,    /* bDescriptorType */
```

```
    /* wTotalLength */
    CONFIG_DESC_TOTAL_LENGTH & 0x00FF,
    (CONFIG_DESC_TOTAL_LENGTH & 0xFF00) >> 8,
    0x01,           /* bNumInterfaces */
    0x01,           /* bConfigurationValue */
    0x00,           /* iConfiguration */
    0x80 | (USBD_SELF_POWERED << 6) | (USBD_REMOTE_WAKEUP << 5),/* bmAttributes */
    USBD_MAX_POWER,         /* MaxPower */

    /* I/F descr: HID */
    LEN_INTERFACE,  /* bLength */
    DESC_INTERFACE, /* bDescriptorType */
    0x00,           /* bInterfaceNumber */
    0x00,           /* bAlternateSetting */
    0x02,           /* bNumEndpoints */
    0x03,           /* bInterfaceClass */
    0x00,           /* bInterfaceSubClass */
    0x00,           /* bInterfaceProtocol */
    0x00,           /* iInterface */

    /* HID Descriptor */
    LEN_HID,                            /* Size of this descriptor in UINT8s. */
    DESC_HID,                           /* HID descriptor type. */
    0x10, 0x01,                         /* HID Class Spec. release number. */
    0x00,                               /* H/W target country. */
    0x01,                               /* Number of HID class descriptors to follow. */
    DESC_HID_RPT,                       /* Descriptor type. */
    /* Total length of report descriptor. */
    sizeof(g_CfuVirtualHid_HidReportDescriptor) & 0x00FF,
    (sizeof(g_CfuVirtualHid_HidReportDescriptor) & 0xFF00) >> 8,
    /* EP Descriptor: interrupt in. */
    LEN_ENDPOINT,                       /* bLength */
    DESC_ENDPOINT,                      /* bDescriptorType */
    (CFU_INT_IN_EP_NUM | EP_INPUT),     /* bEndpointAddress */
    EP_INT,                             /* bmAttributes */
    /* wMaxPacketSize */
    EP2_MAX_PKT_SIZE & 0x00FF,
    (EP2_MAX_PKT_SIZE & 0xFF00) >> 8,
    HID_DEFAULT_INT_IN_INTERVAL,        /* bInterval */
    /* EP Descriptor: interrupt out. */
    LEN_ENDPOINT,                       /* bLength */
    DESC_ENDPOINT,                      /* bDescriptorType */
    (CFU_INT_OUT_EP_NUM | EP_OUTPUT),   /* bEndpointAddress */
    EP_INT,                             /* bmAttributes */
    /* wMaxPacketSize */
    EP3_MAX_PKT_SIZE & 0x00FF,
    (EP3_MAX_PKT_SIZE & 0xFF00) >> 8,
    HID_DEFAULT_INT_IN_INTERVAL,        /* bInterval */
};
```

1.3.2.3 CFU HID 报告描述符

CFU HID 报告描述符记录了重要信息，帮助主机找到目标以执行 CFU 过程。在此描述符中，每个参数都应检查是否与驱动程序匹配，例如用于提案/负载输入和输出的 USAGE_PAGE 和 REPORT_ID。

```
static uint8_t g_CfuVirtualHid_HidReportDescriptor[] =
{
    0x06, CFU_DEVICE_USAGE_PAGE,        // USAGE_PAGE(0xFA00)
    0x09, CFU_DEVICE_USAGE,             // USAGE(0xF5)
    0xA1, 0x01,                         // COLLECTION(0x01)
    0x15, 0x00,                         // LOGICAL_MINIMUM(0)
    0x27, 0xFF, 0xFF, 0xFF, 0xFF,       // LOGICAL_MAXIMUM(-1)

    // Necessary to work around possible bug in HID class related to VHF.

    0x85, REPORT_ID_PAYLOAD_INPUT,      // REPORT_ID(34)
    0x75, INPUT_REPORT_LENGTH,          // REPORT SIZE(32)
    0x95, 0x04,                         // REPORT COUNT(4)
    0x19, PAYLOAD_INPUT_USAGE_MIN,      // USAGE MIN (0x26)
    0x29, PAYLOAD_INPUT_USAGE_MAX,      // USAGE MAX (0x29)
    0x81, 0x02,                         // INPUT(0x02)

    0x85, REPORT_ID_OFFER_INPUT,        // REPORT_ID(37)
    0x75, INPUT_REPORT_LENGTH,          // REPORT SIZE(32)
    0x95, 0x04,                         // REPORT COUNT(4)
    0x19, OFFER_INPUT_USAGE_MIN,        // USAGE MIN (0x1A)
    0x29, OFFER_INPUT_USAGE_MAX,        // USAGE MAX (0x1D)
    0x81, 0x02,                         // INPUT(0x02)

    0x85, REPORT_ID_PAYLOAD_OUTPUT,     // REPORT_ID(33)
    0x75, 0x08,                         // REPORT SIZE(8)
    0x95, OUTPUT_REPORT_LENGTH,         // REPORT COUNT(60)
    0x09, PAYLOAD_OUTPUT_USAGE,         // USAGE(0x31)
    0x92, 0x02, 0x01,                   // OUTPUT(0x02)

    0x85, REPORT_ID_OFFER_OUTPUT,       // REPORT_ID(36)
    0x75, INPUT_REPORT_LENGTH,          // REPORT SIZE(32)
    0x95, 0x04,                         // REPORT COUNT(4)
    0x19, OFFER_OUTPUT_USAGE_MIN,       // USAGE MIN (0x1E)
    0x29, OFFER_OUTPUT_USAGE_MAX,       // USAGE MAX (0x21)
    0x91, 0x02,                         // OUTPUT(0x02)

    0x85, REPORT_ID_VERSIONS_FEATURE,   // REPORT_ID(32)
    0x75, 0x08,                         // REPORT SIZE(8)
    0x95, FEATURE_REPORT_LENGTH,        // REPORT COUNT(60)
    0x09, VERSIONS_FEATURE_USAGE,       // USAGE(0x42)
    0xB2, 0x02, 0x01,                   // FEATURE(0x02)

    0xC0                                // END_COLLECTION()
};
```

1.4 驱动程序信息

在 CFU 过程中，当固件和驱动程序通过必要的测试后，固件提供商可以通过 Windows Update 上传其驱动程序。主计算机可以在日常的 Windows 更新过程中轻松获取驱动程序。

在测试阶段，提供商应该在独立环境中创建、打包并测试其驱动程序。这可以通过使用 Microsoft 提供的范例项目来实现。

建议使用 Microsoft 在 GitHub 上提供的项目来构建驱动程序：microsoft/CFU: Component Firmware Update (github.com)，该项目在 Host 文件夹中包含多种类型的范例。

有关实现和详细信息，请参考 Microsoft 提供的以下连结：Component Firmware Update (CFU) INF Configuration - Windows drivers | Microsoft Learn。

VirtualDevicePackage 文件夹中提供了一个测试用驱动程序范例。尽管在 CFU 过程中主机应从 Windows Update 获取并自动安装驱动程序，但开发人员也可以使用 .inf 档案手动安装和测试驱动程序。

1.5 范例驱动程序安装

范例驱动程序与 CFU 进程中使用的常规驱动程序不同，因为它不会通过 Windows Update 路径。为了测试范例驱动程序是否能够安装并在使用者环境中正常运作，使用者需要按照以下步骤进行。

在安装驱动程序之前，使用者需要注册数字签名。请双击名为 virtualdevicefirmwareupdate.cer 的 .cer 档案，然后点击安装凭证按钮。凭证汇入精灵窗口应会出现。选择证书存储域为本地机器，并在取得管理员许可后点击下一步按钮。接着，在凭证存放页面上，选择将所有凭证放入以下的存放区选项，点击浏览，然后选择受信任的跟证书授权单位以储存该凭证。

在凭证注册完成后，右键点击 virtualdevicefirmwareupdate.inf 档案，选择安装选项。此时 .inf 驱动程序的安装将完成。

1.6 执行结果

首先，确保驱动程序已安装在主计算机上。当主要组件通过 USB 连接到主计算机时，检查装置管理器中的固件部分。如果驱动程序设定与装置匹配，但驱动程序名称未显示或与驱动程序设定不一致，则尝试手动更新驱动程序。

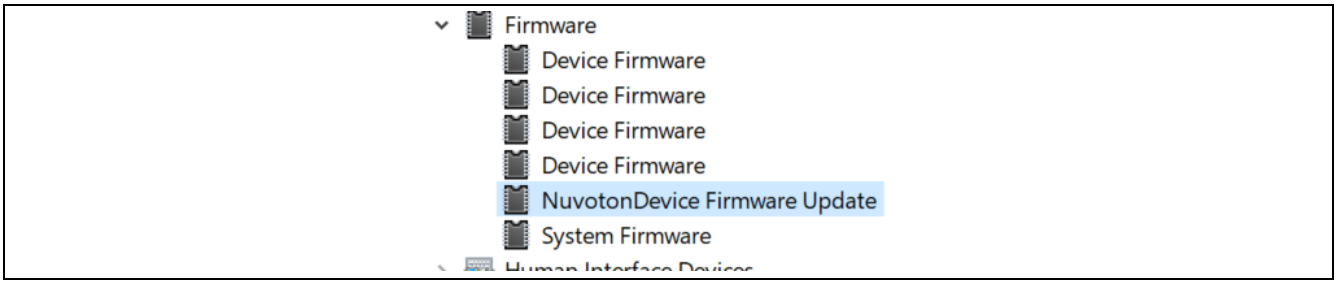
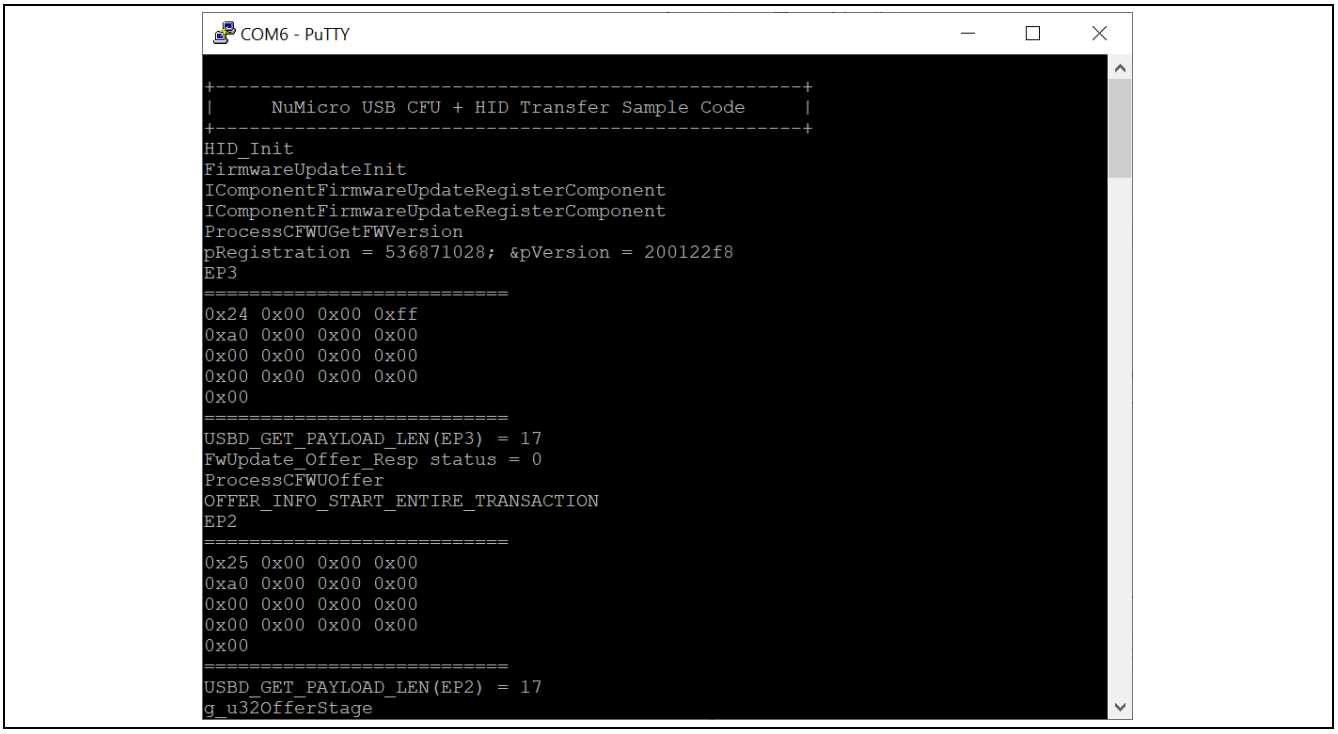


图 1-3 设备管理器

当配置设定正确的开发板连接到主机时，CFU 过程将自动启动。通过在 NuMaker-PFM-M2351 开发板上使用 ICE 连接主机，可以使用 VCOM 检查输出，这将显示主机与主要组件之间的传输信息。



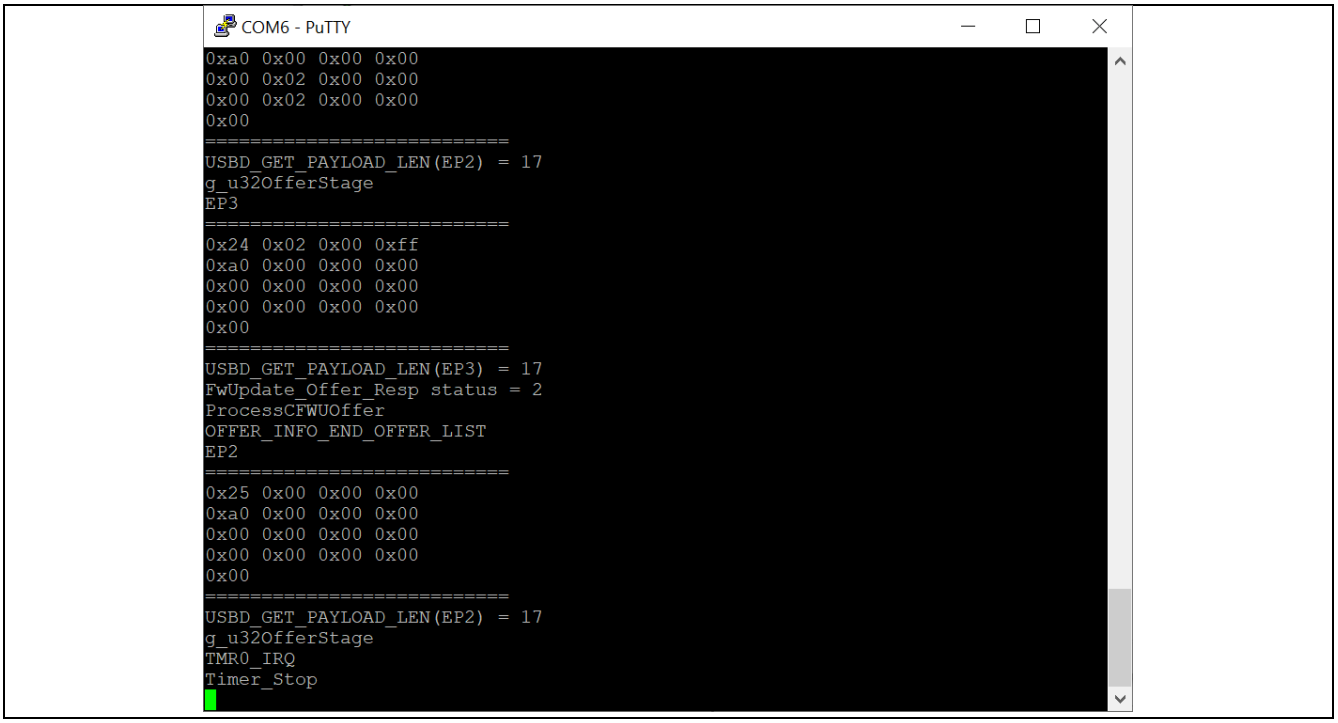


图 1-4 UART 信息

由于主要组件与子组件之间的传输将由使用者自行实现，这些信息只关注 CFU 过程的正确性。此外，设置超时非常重要，否则如果提案的接受/跳过/拒绝的实现出现问题，可能会陷入无限提案的情况。

2. 代码介绍

组件装置范例程序的主程序如下：在初始化系统后，主程序设置 USB 装置，并通过 USB 中断等待来自主机的 CFU 提案发送。

```
int32_t main(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();
    /* Init UART for printf */
    UART0_Init();
    printf("\n");
    printf("+-----------------------------------------------------+\n");
    printf("|     NuMicro USB CFU + HID Transfer Sample Code      |\n");
    printf("+-----------------------------------------------------+\n");
    USBD_Open(&gsInfo, HID_ClassRequest, NULL);
    /* Endpoint configuration */
    HID_Init();
    USBD_Start();
    NVIC_EnableIRQ(USBD_IRQn);
#if CRYSTAL_LESS
    /* Backup default trim */
    s_u32DefaultTrim = M32(TRIM_INIT);
    s_u32LastTrim = s_u32DefaultTrim;
#endif
    /* Clear SOF */
    USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;
    /* Enable FMC ISP function */
    FMC_Open();
    FirmwareUpdateInit();
    // Register the component chain
    IComponentFirmwareUpdateRegisterComponent(&s_Comp_Registration);
    IComponentFirmwareUpdateRegisterComponent(&s_Comp_Registration_2);
    ProcessCFWUGetFWVersion(&s_Get_FwVer_Resp);

    while (1)
    {
#if CRYSTAL_LESS

        /* Start USB trim if it is not enabled. */
        if ((SYS->TCTL48M & SYS_TCTL48M_FREQSEL_Msk) != 1)
        {
            /* Start USB trim only when SOF */
            if (USBD->INTSTS & USBD_INTSTS_SOFIF_Msk)
            {
                /* Clear SOF */
                USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;
                /* Re-enable crystal-less */
                SYS->TCTL48M = 0x01;
                SYS->TCTL48M |= SYS_TCTL48M_REFCKSEL_Msk;
            }
```

```
        }

        /* Disable USB Trim when error */
        if (SYS->TISTS48M & (SYS_TISTS48M_CLKERRIF_Msk | SYS_TISTS48M_TFAILIF_Msk))
        {
            /* Last TRIM */
            M32(TRIM_INIT) = s_u32LastTrim;
            /* Disable crystal-less */
            SYS->TCTL48M = 0;
            /* Clear error flags */
            SYS->TISTS48M = SYS_TISTS48M_CLKERRIF_Msk | SYS_TISTS48M_TFAILIF_Msk;
            /* Clear SOF */
            USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;
        }

        /* Check trim value whether it is over the threshold */
        if ((M32(TRIM_INIT) > (s_u32DefaultTrim + TRIM_THRESHOLD)) ||
                (M32(TRIM_INIT) < (s_u32DefaultTrim - TRIM_THRESHOLD)))
        {
            /* Write updated value */
            M32(TRIM_INIT) = s_u32LastTrim;
        }
        else
        {
            /* Backup trim value */
            s_u32LastTrim = M32(TRIM_INIT);
        }

#endif
    }
}
```

在范例程序中，端点 2 和端点 3 负责处理 HID CFU 通讯的输入和输出。

```
void EP2_Handler(void)  /* Interrupt IN handler */
{
    uint8_t *pu8Ptr;
    CFU_SetInReport();
}

void EP3_Handler(void)  /* Interrupt OUT handler */
{
    uint8_t *pu8Ptr;
    printf("EP3\n");
    printf("============================\n");
    /* Interrupt OUT */
    pu8Ptr = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP3));

    for (int i = 0; i < USBD_GET_PAYLOAD_LEN(EP3); i ++)
    {
        printf("0x%02x ", pu8Ptr[i]);

        if (i % 4 == 3 || i == USBD_GET_PAYLOAD_LEN(EP3) - 1) printf("\n");
    }

    printf("============================\n");
```

```
    printf("USBD_GET_PAYLOAD_LEN(EP3) = %d\n", USBD_GET_PAYLOAD_LEN(EP3));
    CFU_GetOutReport(pu8Ptr + 1, USBD_GET_PAYLOAD_LEN(EP3) - 1);
    USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
}
```

为了使程序在接收到 CFU 数据时能够继续进行，每个组件应通过使用 **COMPONENT_REGISTRATION** 结构将其信息注册到注册链中。该结构包含组件 ID、必要功能的指针，以及下一个注册的指标。

```
typedef struct COMPONENT_REGISTRATION_STRUCT
{
    struct COMPONENT_REGISTRATION_STRUCT *pNext;
    const ICOMPONENT_INTERFACE interface;
    const UINT8 componentId;
} COMPONENT_REGISTRATION;

typedef struct
{
    MCU_STATUS(*GetVersion)(UINT32* pVersion);
    MCU_STATUS(*GetProductInfo)(UINT32* pProductInfo);
    MCU_STATUS(*ProcessOffer)(FWUPDATE_OFFER_COMMAND* pCommand,
                              FWUPDATE_OFFER_RESPONSE* pResponse);
    MCU_STATUS(*GetCrcOffset)(UINT32* pOffset);
    MCU_STATUS(*NotifySuccess)(BOOL forceReset,
                               READ_FIRMWARE_FUNC readHandler,
                               READ_COMPLETED_FUNC readCompleteHandler);
} ICOMPONENT_INTERFACE;
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M2351_Series_BSP_CMSIS_V3.00.006
- 驱动程序
  - 位于文件夹 /VirtualDevicePackage 的范例驱动程序或自有驱动程序

3.2 硬件需求

- 电路组件
  - NuMaker-PFM-M2351 V1.0
    - ♦ USB 连接到 USB 1.1 作为主程序使用
    - ♦ UART 连接到 ICEV3.0 (VCOM) 用于信息打印

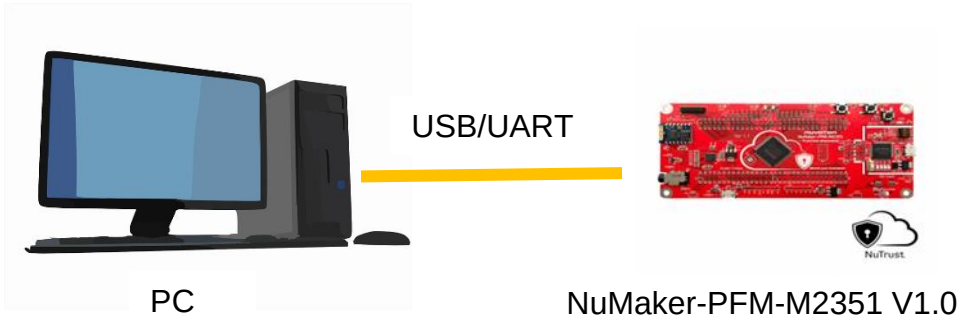


图 3-1 USB CFU 连接执行图

4. 目录信息

目录结构如下图所示。

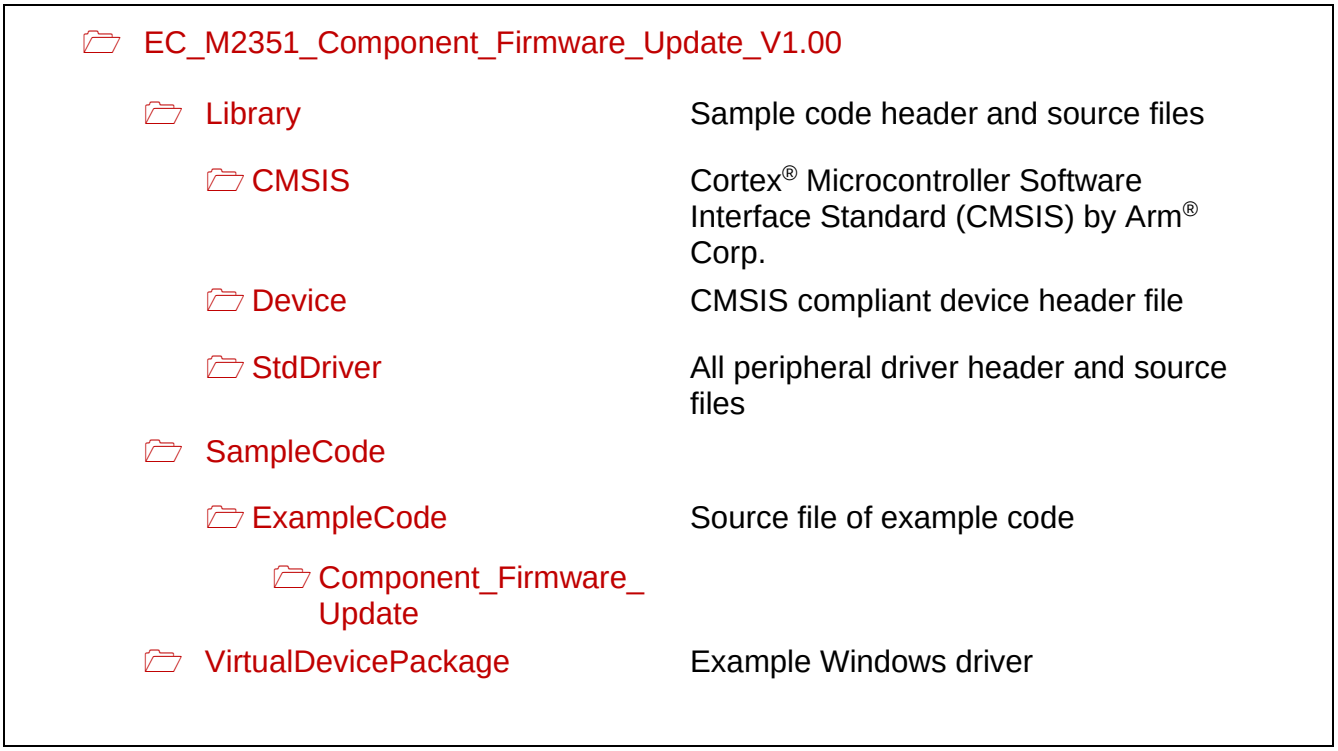


图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *Component_Firmware_Update.uvprojx*
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码
   - 通过 UART 界面显示结果

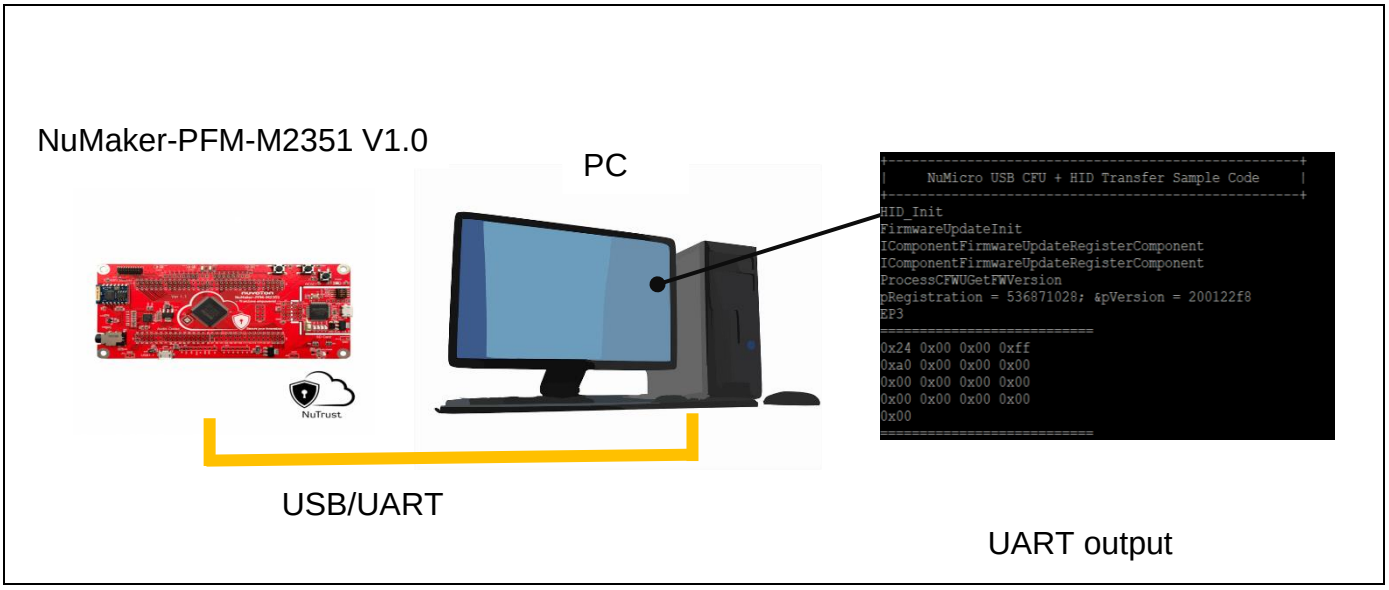


图 5-1 USB CFU 结果

6. 修订纪录

Date	Revision	Description
2024.11.21	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.