M2351 USB 元件韌體更新

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

應用簡述	此範例程式展示了在 M2351 開發平台上實現元件韌體更新 (CFU) 功能的方式。
BSP 版本	M2351_Series_BSP_CMSIS_V3.00.006
開發平台	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

此範例程式使用 NuMaker-PFM-M2351 開發平台，展示一個支援元件韌體更新 (CFU) 的 USB 裝置。當安裝了來自 Windows Update 的更新資訊的電腦連接到該 USB 裝置時，CFU 將啟動，開始更新該 USB 裝置的子元件。

1.1 NuMaker-PFM-M2351 開發平台

NuMaker-PFM-M2351 是一款基於 NuMicro®-M2351 微控制器的開發板。Nuvoton NuMicro® M2351 安全微控制器系列採用 Arm® Cortex®-M23 核心，並搭載適用於 Armv8-M 架構的 TrustZone® 技術，可將傳統的韌體安全性提升至更高層次的強大軟體安全性。

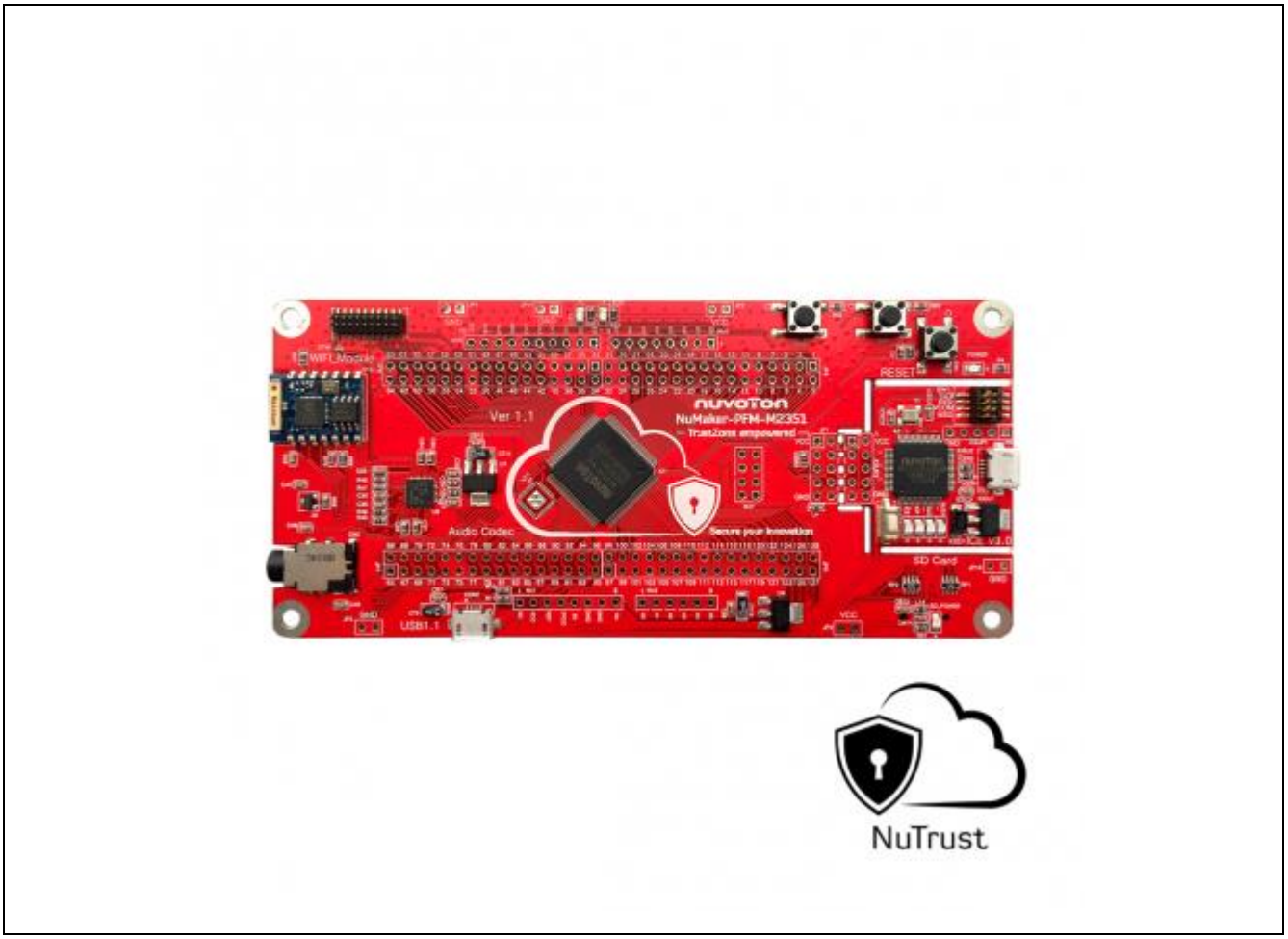


圖 1-1 NuMaker-PFM-M2351 V1.0 開發平台

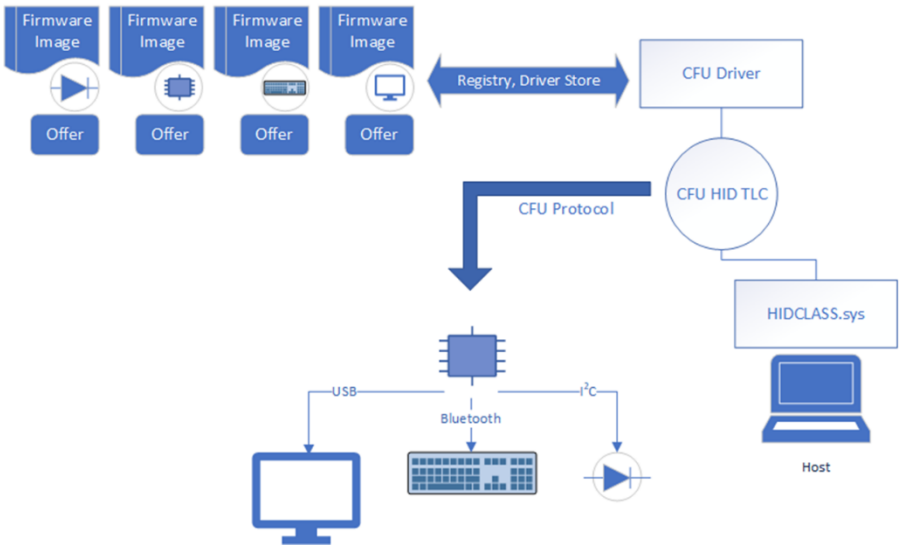
1.2 CFU 工作原理分析

1.2.1 背景介紹

元件韌體更新 (CFU) 是 Microsoft 提供的一項服務，用於自動更新硬體韌體，以提升裝置的性能和安全性。CFU 的更新過程幾乎不會打擾用戶，甚至無需進入「更新模式」，用戶無需等待或意識到更新正在進行。此外，CFU 基本上是自動更新：系統會自動檢測並安裝最新的韌體更新，而無需用戶干預。.

CFU 是透過 Windows Update 驅動程式提供的。而目標裝置必須能夠在下載不適合的韌體之前將其拒絕。

1.2.2 CFU 進程開始前的示意圖



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

圖 1-2 架構示意圖

1. 韌體提供商透過 Windows Update 上傳其驅動程式。
2. 客戶端在主機端接收驅動程式。
3. 當與驅動程式設定相符的目標裝置連接到主機時，主機將啟動 CFU 進程。
4. 在 CFU 進程中，子元件的韌體將透過接收來自主要目標元件的資料進行更新。

1.2.3 傳輸數據格式詳情

CFU 提交給常駐韌體的資料是一對檔案，其中一個是「提案部分」，另一個是「內容部分」。CFU 驅動程式（主機）可能包含主要元件及其相關子元件的多個韌體映像檔。

提案部分包含足夠的負載資訊，以便主要元件決定該負載是否可接受。

提案資訊包括 CFU 協議版本、元件 ID（如果適用，也包含子元件 ID）、韌體版本、釋出版本與除錯版本狀態，以及其他相關資訊。

封包的負載基本上是一組地址和固定大小的位元組陣列，用於進行編程操作。這些位元組對主機而言是不可見的。

1.2.4 CFU 過程詳情

使用 CFU 進行韌體更新的一般流程是由主機向主要元件發送每個封包的提案。

當主機檢測到與韌體匹配的裝置已連接時，主機會向裝置發送「主機初始化通知」，告知裝置 CFU 更新過程已準備就緒並等待接收。隨後，主機會發送「提案開始通知」。

接著，主機將按順序發送元件的提案。對於每個獨立的元件提案，主要元件可以選擇接受、拒絕或跳過該提案。如果提案被接受，主機將開始發送更新數據並啟動更新過程；否則，主機將跳過並依次發送其他元件的提案。

這個過程將持續進行，直到裝置對提案鏈中的每個元件提案都返回「拒絕」時為止。然後主機會發送「提案結束通知」，並完成 CFU 過程。

1.3 USB 描述符分析

1.3.1 元件 USB 描述符概述

USB 描述符是一組結構化數據，用於描述 USB 裝置的屬性、功能和需求。這些描述符幫助主機識別裝置類型、可用功能以及傳輸需求。

1.3.2 描述符在元件裝置中的作用

在處理 CFU 過程的元件裝置中，描述符提供了裝置如何與主機交互以及傳輸數據的詳細資訊。以下概述了每個描述符在視頻傳輸中的具體作用及其在程式中的實現。

1.3.2.1 裝置描述符

裝置描述符提供有關 USB 裝置的基本資訊，例如裝置類型、廠商 ID、產品 ID 以及支援的 USB 版本。這是主機獲取裝置資訊的第一步，使主機能夠識別裝置的基本屬性。

```
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

配置描述符描述裝置的配置選項，包括電源使用情況和支援的介面數量。主機利用此描述符來配置裝置，以確保其正常運作。該描述符還包含 HID 描述符、介面描述符和端點描述符。

介面描述符 提供了關於配置中每個介面的詳細資訊，包括它們提供的功能（例如視頻控制或視頻流）以及使用的端點數量。

端點描述符 定義主機與裝置之間數據傳輸的具體方法。對於視頻流，端點通常配置為同步傳輸，以確保數據穩定性和實時傳輸。在範例程式碼中，使用了端點 2 和端點 3，因此它們的信息必須包含在端點描述符中。

程式碼實現：

```
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 報告描述符

CFU HID 報告描述符記錄了重要資訊，幫助主機找到目標以執行 CFU 過程。在此描述符中，每個參數都應檢查是否與驅動程式匹配，例如用於提案/負載輸入和輸出的 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 驅動程式資訊

在 CFU 過程中，當韌體和驅動程式通過必要的測試後，韌體提供商可以透過 Windows Update 上傳其驅動程式。主機電腦可以在日常的 Windows 更新過程中輕鬆獲取驅動程式。

在測試階段，提供商應該在獨立環境中創建、打包並測試其驅動程式。這可以透過使用 Microsoft 提供的範例專案來實現。

建議使用 Microsoft 在 GitHub 上提供的專案來構建驅動程式：microsoft/CFU: Component Firmware Update (github.com)，該專案在 Host 資料夾中包含多種類型的範例。

有關實現和詳細資訊，請參考 Microsoft 提供的以下連結：Component Firmware Update (CFU) INF Configuration - Windows drivers | Microsoft Learn。

VirtualDevicePackage 資料夾中提供了一個測試用驅動程式範例。儘管在 CFU 過程中主機應從 Windows Update 獲取並自動安裝驅動程式，但開發人員也可以使用 .inf 檔案手動安裝和測試驅動程式。

1.5 範例驅動程式安裝

範例驅動程式與 CFU 進程中使用的常規驅動程式不同，因為它不會通過 Windows Update 路徑。為了測試範例驅動程式是否能夠安裝並在使用者環境中正常運作，使用者需要按照以下步驟進行。

在安裝驅動程式之前，使用者需要註冊數位簽章。請雙擊名為 virtualdevicefirmwareupdate.cer 的 .cer 檔案，然後點擊安裝憑證按鈕。憑證匯入精靈視窗應會出現。選擇憑證存放區域為本地機器，並在取得管理員許可後點擊下一步按鈕。接著，在憑證存放頁面上，選擇將所有憑證放入以下的存放區選項，點擊瀏覽，然後選擇受信任的根憑證授權單位以儲存該憑證。

在憑證註冊完成後，右鍵點擊 virtualdevicefirmwareupdate.inf 檔案，選擇安裝選項。此時 .inf 驅動程式的安裝將完成。

1.6 執行結果

首先，確保驅動程式已安裝在主機電腦上。當主要元件透過 USB 連接到主機電腦時，檢查裝置管理器中的韌體部分。如果驅動程式設定與裝置匹配，但驅動程式名稱未顯示或與驅動程式設定不一致，則嘗試手動更新驅動程式。

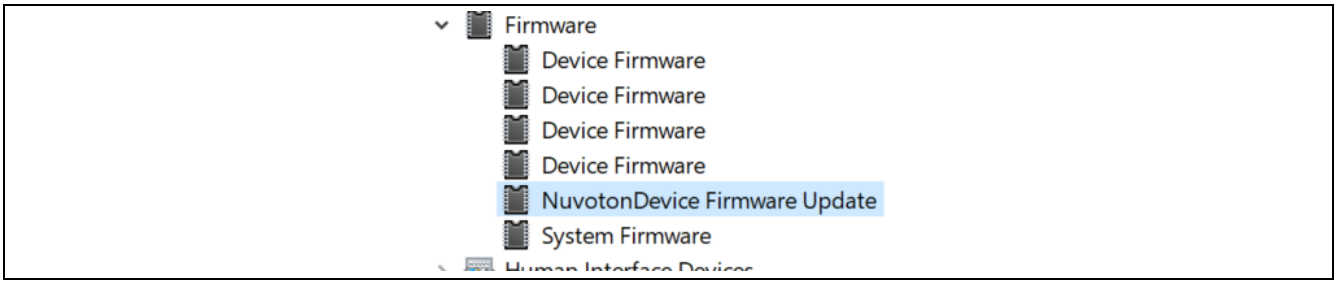
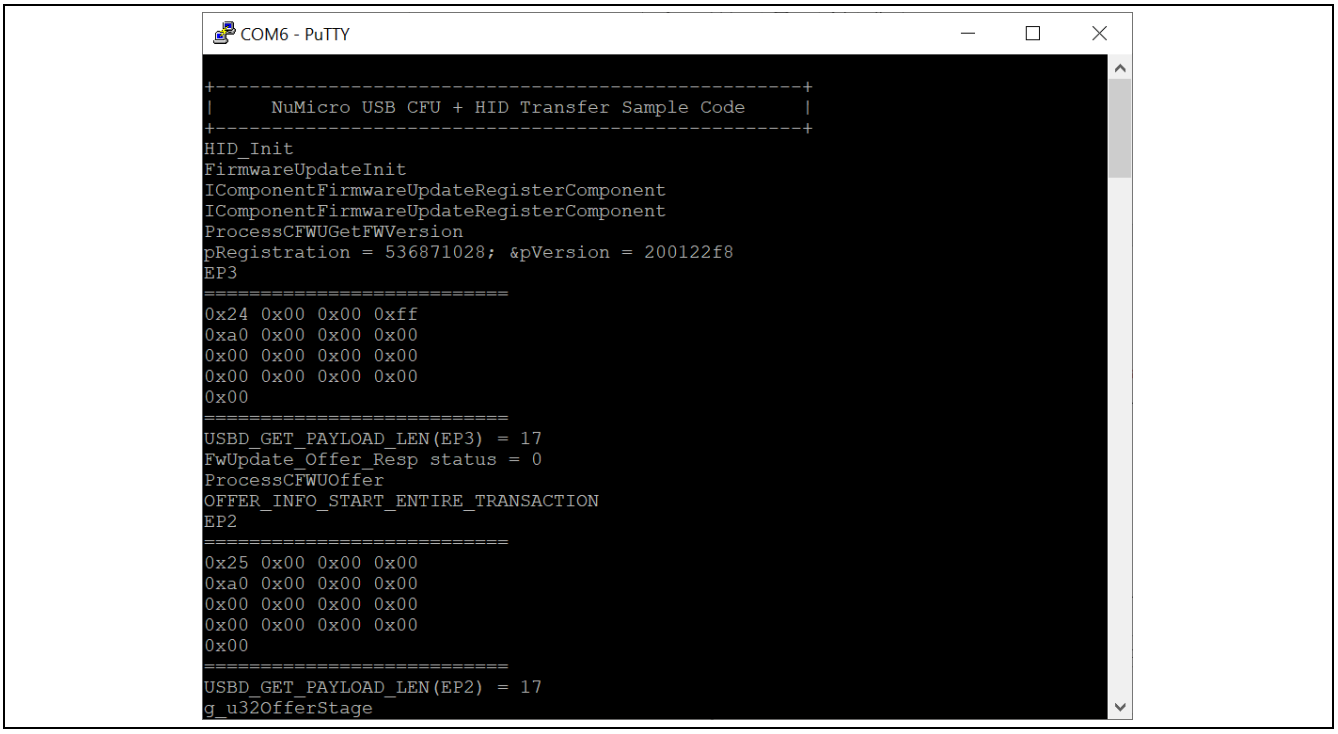


圖 1-3 裝置管理員

當配置設定正確的開發板連接到主機時，CFU 過程將自動啟動。透過在 NuMaker-PFM-M2351 開發板上使用 ICE 連接主機，可以使用 VCOM 檢查輸出，這將顯示主機與主要元件之間的傳輸資訊。



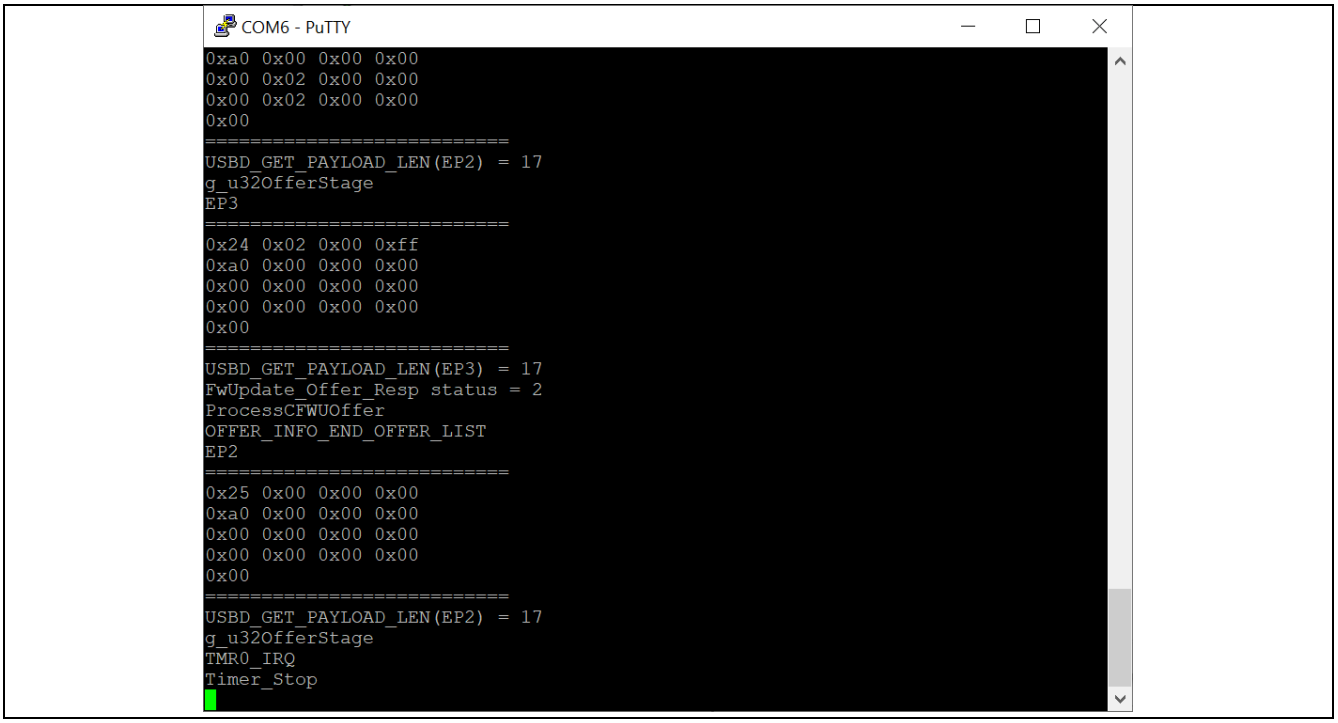


圖 1-4 UART 訊息

由於主要元件與子元件之間的傳輸將由使用者自行實現，這些訊息只關注 CFU 過程的正確性。此外，設置超時非常重要，否則如果提案的接受/跳過/拒絕的實現出現問題，可能會陷入無限提案的情況。

2. 代碼介紹

元件裝置範例程式的主程式如下：在初始化系統後，主程式設置 USB 裝置，並透過 USB 中斷等待來自主機的 CFU 提案發送。

```
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
    }
}
```

在範例程式中，端點 2 和端點 3 負責處理 HID CFU 通訊的輸入和輸出。

```
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);
}
```

為了使程式在接收到 CFU 資料時能夠繼續進行，每個元件應透過使用 **COMPONENT_REGISTRATION** 結構將其資訊註冊到註冊鏈中。該結構包含元件 ID、必要功能的指標，以及下一個註冊的指標。

```
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. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M2351_Series_BSP_CMSIS_V3.00.006
- 驅動程式
  - 位於資料夾 /VirtualDevicePackage 的範例驅動程式或自有驅動程式

3.2 硬體需求

- 電路元件
  - NuMaker-PFM-M2351 V1.0
    - ♦ USB 連接到 USB 1.1 作為主程式使用
    - ♦ UART 連接到 ICEV3.0 (VCOM) 用於訊息打印

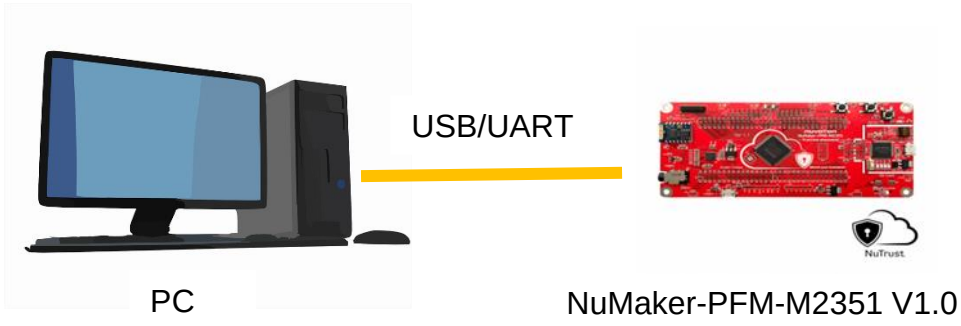


圖 3-1 USB CFU 連接執行圖

4. 目錄資訊

目錄結構如下圖所示。

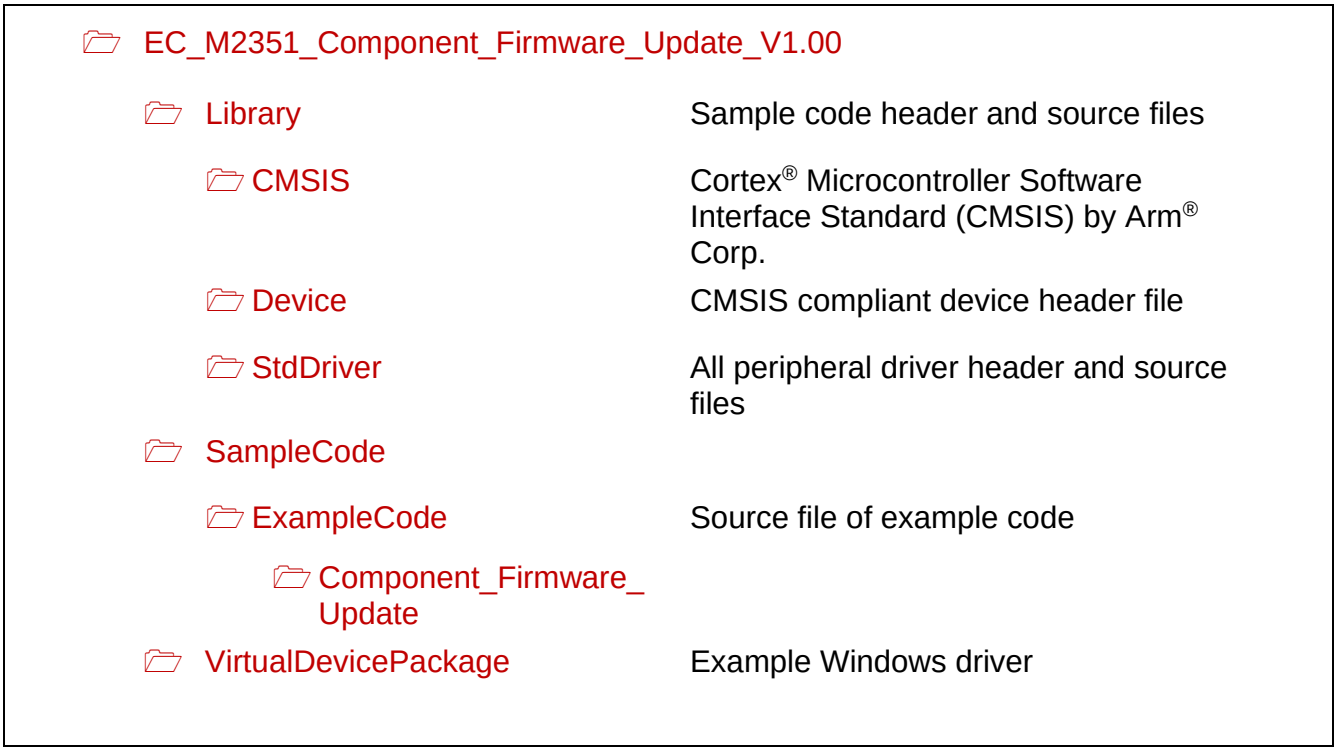


圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *Component_Firmware_Update.uvprojx*
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
2. 進入除錯模式介面
   - 執行代碼
   - 透過 UART 介面顯示結果

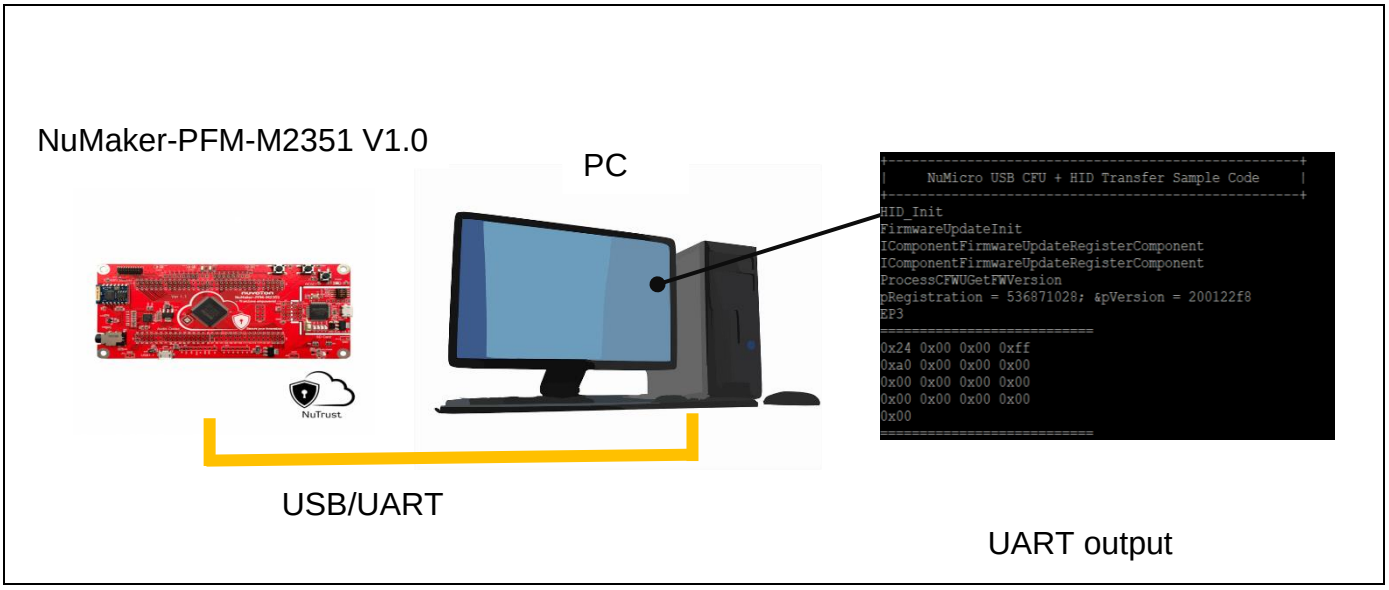


圖 5-1 USB CFU 結果

6. 修訂紀錄

Date	Revision	Description
2024.11.21	1.00	初始發布。

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.