Android ISP

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

代碼簡述	使用 NUC240 的 USB ISP 功能與 Android APP 作韌體升級
BSP 版本	NUC230/240 Series BSP v3.01.002
開發平台	NuTiny–EVB–NUC240–LQFP100 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 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 功能介紹

1.1 簡介

此範例程式演示如何在Android裝置端透過USB OTG(On-The-Go)的連接來控制NUC240端的flash做韌體的升級。

1.2 原理

USB OTG是USB2.0規格的補充標準。 它可使USB設備，例如播放器或手機作為主機，與USB隨身碟、滑鼠或鍵盤連接。此範例程式使用手機(Android設備)作為主機，透過USB OTG與NUC240連接，如圖1-1所示。將NuvotonISPTool應用程式安裝於手機中，並以該應用程式做為控制介面，透過USB OTG來控制NUC240的Flash記憶體的擦除、寫入與存取，以完成韌體的升級。

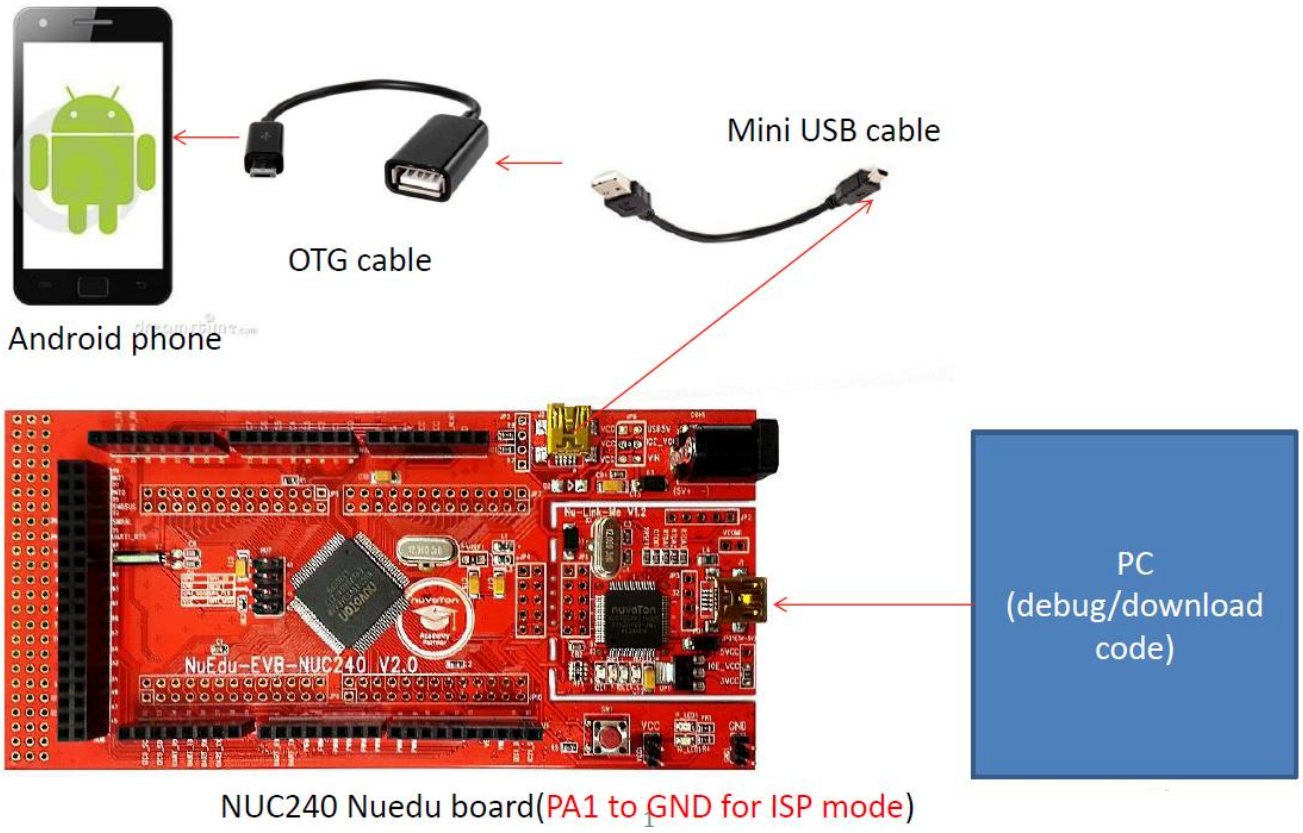

圖 1-1 Android 裝置與 NUC240 的連線與傳輸控制

1.3 執行結果

執行此範例程式前，需先在Android作業系統的手機上安裝NuvotonISPTool。NuvotonISPTool放置的目錄如下：

EC_NUC240_ANDROID_ISP_V1.00\SampleCode\NuAndroidISPTool\

將該目錄中的壓縮檔Examples_04_01.zip檔解壓縮後，會產生一個APK (Android Application Package)的目錄如圖1-2：

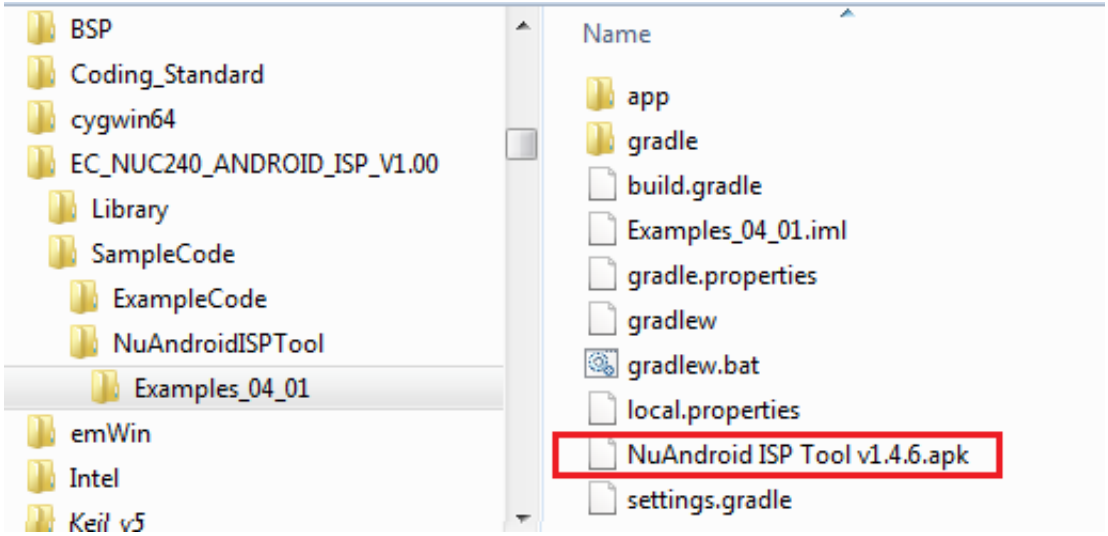

圖1-2 Android application package目錄

將目錄中的NuAndroid ISP Tool v1.4.6.apk下載到手機上並安裝完成後，回到PC端執行範例程式。首先在PC端先開啟專案ANDROID_ISP.uvproj，編譯好範例程式後，將執行程式載入NUC240，並按下"Start/Stop Debug Session"功能鍵，如圖1-3中紅框所示，以進入除錯模式。

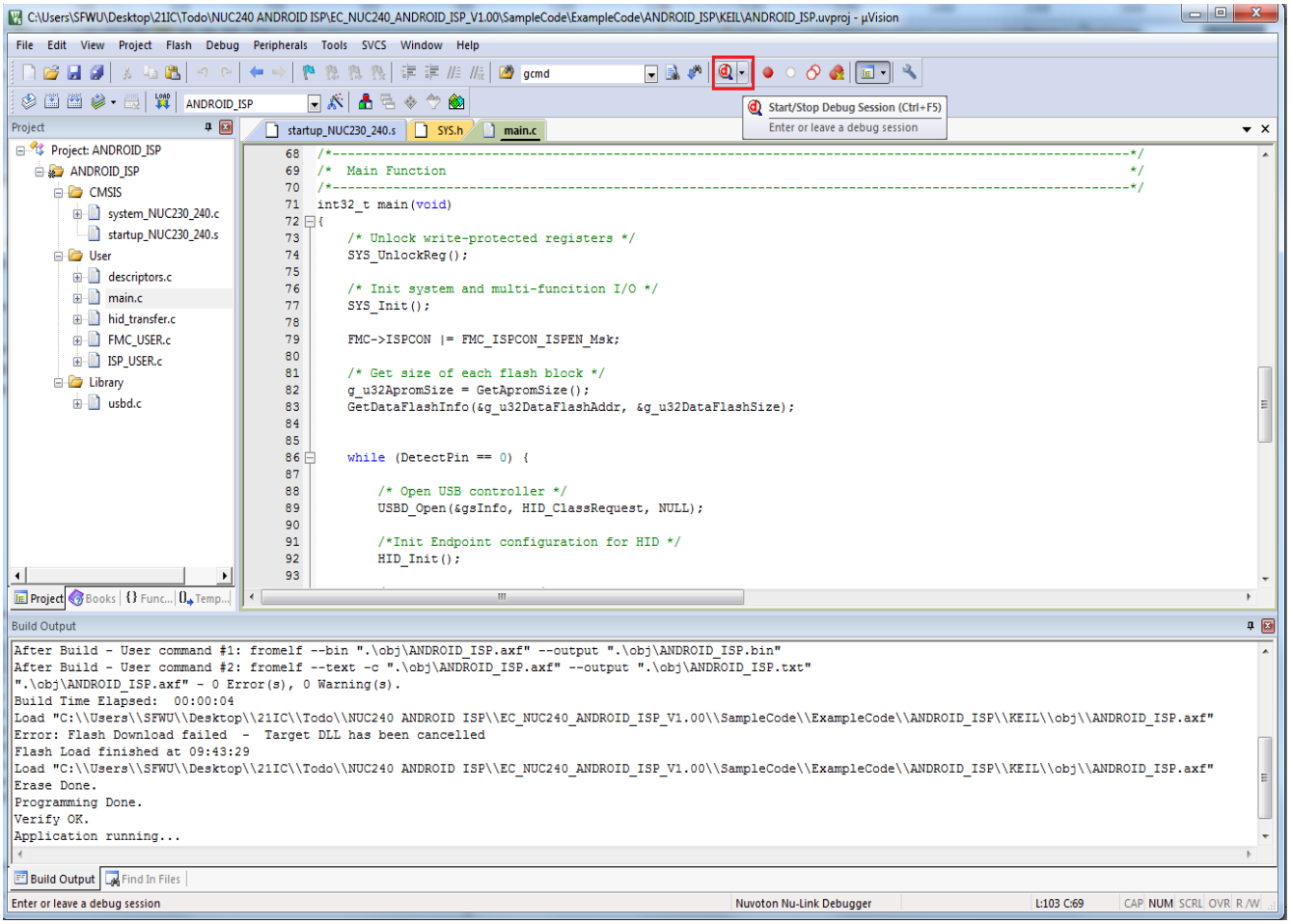

圖1-3 在程式載入NUC240後，進入除錯模式之功能鍵

進入除錯模式後，按下”Run”功能鍵，如圖1-4中紅框所示，開始執行程式以啟動USB連線，以等待Android 裝置的連接。

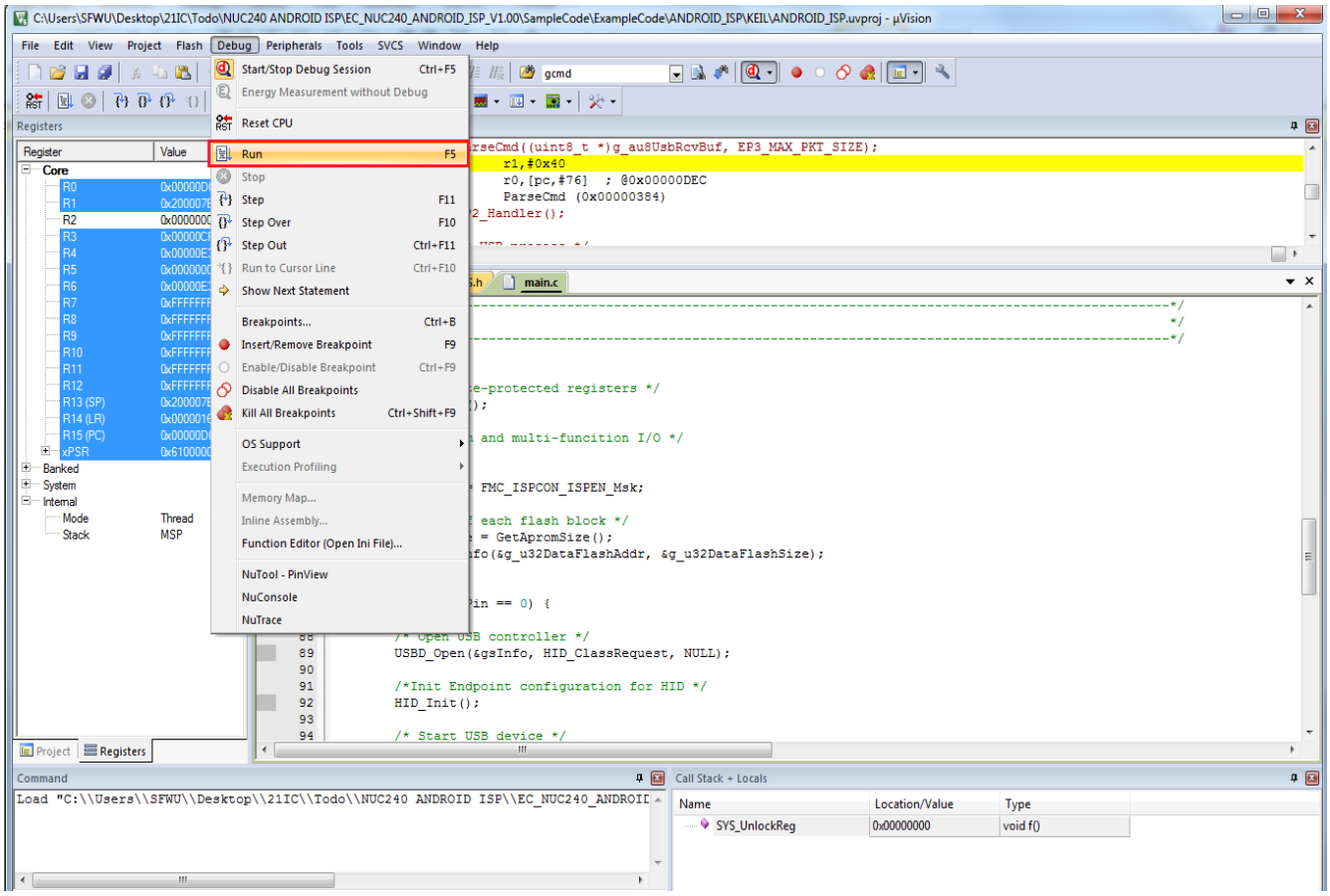

圖1-4 進入除錯模式，執行程式以啟動USB連線

而後以OTG cable和Mini USB cable將Android手機與NUC240做連接，如章節1-2中的圖1-1所示。當手機與NUC240連接完成後，在手機上會自動出現NuvotonISPTool的初始畫面，並顯示USB連線訊息，如圖1-5所示：

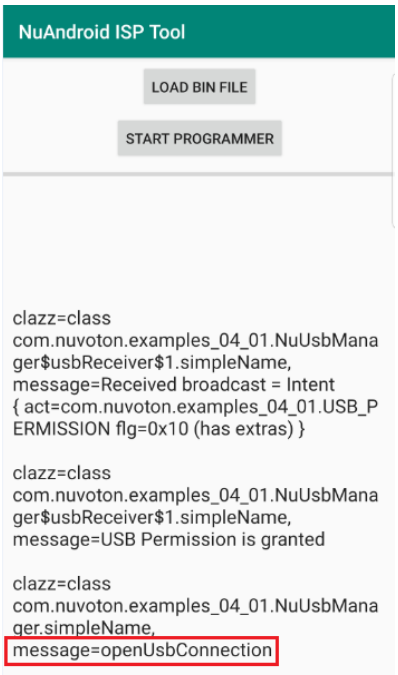

圖 1-5 NuvotonISPTool 啟動畫面與 USB 連線訊息

NuvotonISPTool 開啟後，按下”LOAD BIN FILE”功能鍵，以執行韌體更新，如圖 1-6 所示：

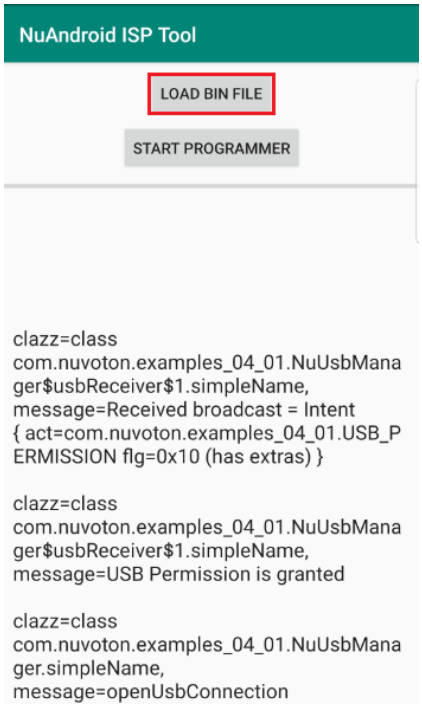

圖 1-6 NuvotonISPTool 執行載入 BIN 檔

按下”LOAD BIN FILE”功能鍵後，進入檔案資料夾，選取欲寫入的 BIN 檔，如圖 1-7 所示：

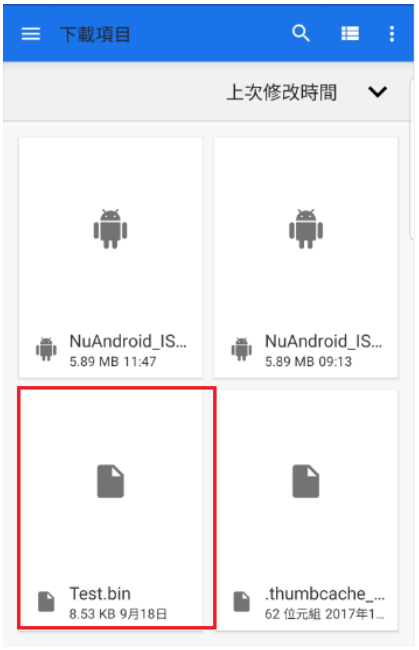

圖 1-7 NuvotonISPTool 選擇欲載入的 BIN 檔

選擇完 BIN 檔之後，回到 NuvotonISPTool 的主畫面，主畫面換顯示 BIN 檔的資訊，如圖 1-8 所示：

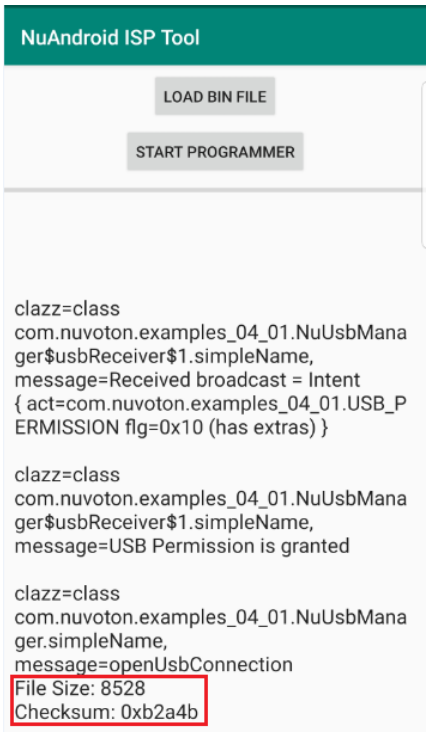

圖 1-8 NuvotonISPTool 顯示 BIN 檔資訊

按下”START PROGRAMMER” 功能鍵，以將所選取的 BIN 檔燒錄至 NUC240 的 flash，如圖 1-9 所示：

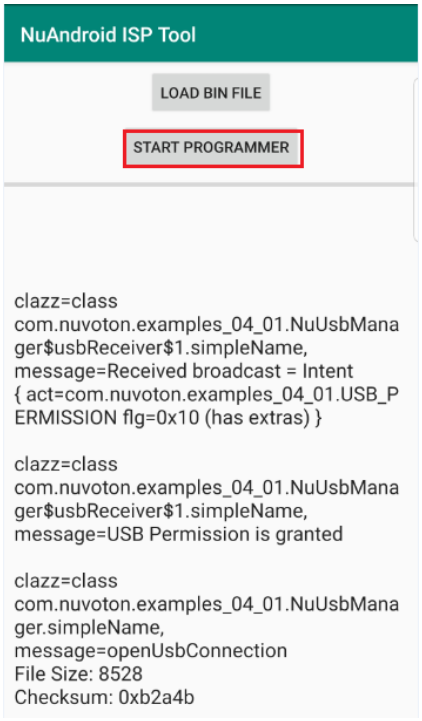

圖 1-9 NuvotonISPTool 執行燒錄 BIN 檔

燒錄完成後，NuvotonISPTool 主畫面會顯示燒錄成功的訊息，如圖 1-10 所示：

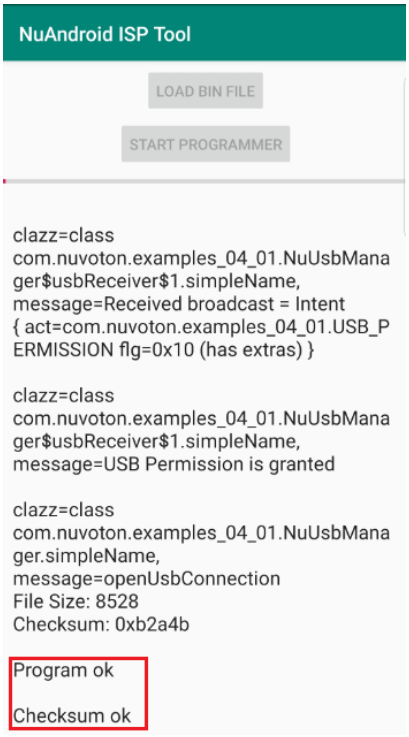

圖 1-10 NuvotonISPTool 燒錄成功訊息

2 代碼介紹

下方的函式為主程式，主要為 USB 等待連線的流程，以及 USB 連線成功後，接收由 Android 手機端的指令以執行 Flash 的讀寫：

```
/*---------------------------------------------------------------------------*/
/*  Main Function                                                            */
/*---------------------------------------------------------------------------*/
int32_t main(void)
{
    /* Unlock write-protected registers */
    SYS_UnlockReg();

    /* Init system and multi-funcition I/O */
    SYS_Init();

    FMC->ISPCON |= FMC_ISPCON_ISPEN_Msk;

    /* Get size of each flash block */
    g_u32ApromSize = GetApromSize();
    GetDataFlashInfo(&g_u32DataFlashAddr, &g_u32DataFlashSize);


    while (DetectPin == 0) {

        /* Open USB controller */
        USBD_Open(&gsInfo, HID_ClassRequest, NULL);

        /*Init Endpoint configuration for HID */
        HID_Init();

        /* Start USB device */
        USBD_Start();

        /* Enable USB device interrupt */
        NVIC_EnableIRQ(USBD_IRQn);

        while (DetectPin == 0) {
            if(bUsbDataReady == TRUE) {
```

```
                /*Parse command from USB transfer*/
                ParseCmd((uint8_t *)g_au8UsbRcvBuf, EP3_MAX_PKT_SIZE);
                EP2_Handler();

                /* USB process */
                bUsbDataReady = FALSE;
            }
        }

        goto _APROM;
    }

    SysTick->LOAD = 300000 * CyclesPerUs;
    SysTick->VAL  = (0x00);
    SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk;

    /* Waiting for down-count to zero */
    while((SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk) == 0);

_APROM:

    /* clear bit */
    outpw(&SYS->RSTSRC, 3);
    outpw(&FMC->ISPCON, inpw(&FMC->ISPCON) & 0xFFFFFFFC);
    outpw(&SCB->AIRCR, (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ));

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

下方的函式為剖析 NUC240 透過 USB 接收的指令，包含 USB 連線，flash 的擦除、讀寫、、等：

```
/*---------------------------------------------------------------------------*/
/*  Command Parsing                                                          */
/*---------------------------------------------------------------------------*/
int ParseCmd(unsigned char *pucBuffer, uint8_t u8Len)
{
    static uint32_t u32StartAddress, u32StartAddressBak, u32TotalLen, u32TotalLenBak,
u32LastDataLen, u32GPackNo = 1;
    uint8_t *pu8Response;
    uint16_t u32LckSum;
    uint32_t u32Lcmd, u32SrcLen, i, u32Regcnf0, u32Security;
```

```
    unsigned char *pucSrc;
    static uint32_t   u32Gcmd;

    pu8Response = g_au8ResponseBuff;

    /* Point to command position in USB transferring packet */
    pucSrc = pucBuffer;
    u32SrcLen = u8Len;
    /* Get command from USB*/
    u32Lcmd = inpw(pucSrc);

    outpw(pu8Response + 4, 0);

    /* Point to PACKNO in USB transferring packet */
    pucSrc += 8;
    u32SrcLen -= 8;

    /* Read config */
    ReadData(CONFIG0, CONFIG0 + 8, (uint32_t*)(pu8Response + 8));
    u32Regcnf0 = *(uint32_t*)(pu8Response + 8);
    u32Security = u32Regcnf0 & 0x2;


    if(u32Lcmd == CMD_SYNC_PACKNO) {
        /* Get PACKNO */
        u32GPackNo = inpw(pucSrc);
    }

    if((u32Lcmd) && (u32Lcmd!=CMD_RESEND_PACKET))
        u32Gcmd = u32Lcmd;

    if(u32Lcmd == CMD_GET_FWVER) {
        /* version 2.3 */
        pu8Response[8] = FW_VERSION;
    } else if(u32Lcmd == CMD_GET_DEVICEID) {
        /* Get device ID */
        outpw(pu8Response+8, SYS->PDID);
        goto out;
    } else if(u32Lcmd == CMD_RUN_APROM || u32Lcmd == CMD_RUN_LDROM || u32Lcmd ==
CMD_RESET) {

        /* clear bit */
```

```
        outpw(&SYS->RSTSRC, 3);
        /* Set Boot Selection */
        if(u32Lcmd == CMD_RUN_APROM) {
            /* Boot from APROM */
            i = (FMC->ISPCON & 0xFFFFFFFC);
        } else if(u32Lcmd == CMD_RUN_LDROM) {
            /* Boot from LDROM */
            i = (FMC->ISPCON & 0xFFFFFFFC);
            i |= 0x00000002;
        } else {
            /* ISP disable */
            i = (FMC->ISPCON & 0xFFFFFFFE);
        }

        outpw(&FMC->ISPCON, i);
        outpw(&SCB->AIRCR, (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ));

        /* Trap the CPU */
        while(1);
    } else if(u32Lcmd == CMD_CONNECT) {
        /* Execute connection */
        u32GPackNo = 1;
        goto out;
    } else if(u32Lcmd == CMD_DISCONNECT) {
        /* Execute disconnection */
        return 0;
    }

    else if((u32Lcmd == CMD_UPDATE_APROM) || (u32Lcmd == CMD_ERASE_ALL)) {
        if(u32Lcmd == CMD_ERASE_ALL) {
            /* Erase APROM + data flash */
            /* Erase all aprom including data flash */
            EraseAP(FALSE, 0, (g_u32ApromSize < 0x20000)?0x20000:g_u32ApromSize);
            *(uint32_t*)(pu8Response + 8) = u32Regcnf0|0x02;
            UpdateConfig((uint32_t*)(pu8Response + 8), NULL);
        } else
            /* Don't erase data flash */
            EraseAP(TRUE, 0, 0);
        g_u32UpdateApromCmd = TRUE;
    }
#ifdef SUPPORT_WRITECKSUM
```

```
    else if(u32Lcmd == CMD_WRITE_CHECKSUM) {
        /* write cksum to aprom last */
        uint32_t cktotallen = inpw(pucSrc);
        u32LckSum = inpw(pucSrc+4);
        CheckCksumBase();
        ReadData(g_ckbase & 0xFFE00, g_ckbase, (uint32_t*)s_au8ApromBuf);
        outpw(s_au8ApromBuf+FMC_FLASH_PAGE_SIZE - 8, cktotallen);
        outpw(s_au8ApromBuf+FMC_FLASH_PAGE_SIZE - 4, u32LckSum);
        FMC_Erase(g_ckbase & 0xFFE00);
        WriteData(g_ckbase & 0xFFE00, g_ckbase + 8, (uint32_t*)s_au8ApromBuf);
    }
#endif
    else if(u32Lcmd == CMD_GET_FLASHMODE) {
        /* return 1: APROM, 2: LDROM */
        outpw(pu8Response+8, (FMC->ISPCON&0x2)? 2 : 1);
    }


    if((u32Lcmd == CMD_UPDATE_APROM) || (u32Lcmd == CMD_UPDATE_DATAFLASH)) {
        /* Update APROM or data flash */
        if(u32Lcmd == CMD_UPDATE_DATAFLASH) {
            /* Update data flash, start form data flash position */
            u32StartAddress = g_u32DataFlashAddr;

            if(g_u32DataFlashSize) {
                EraseAP(FALSE, g_u32DataFlashAddr, g_u32DataFlashAddr +
g_u32DataFlashSize);
            } else
                goto out;
        } else {
            /* Update APROM, start form 0x0 */
            u32StartAddress = 0;
        }

        /* Get write size */
        u32TotalLen = inpw(pucSrc+4);
        pucSrc += 8;
        u32SrcLen -= 8;
        /* Backup start address */
        u32StartAddressBak = u32StartAddress;
        /* Backup write size */
        u32TotalLenBak = u32TotalLen;
```

```
    } else if(u32Lcmd == CMD_UPDATE_CONFIG) {
        /* security lock */
        if((u32Security == 0) && (!g_u32UpdateApromCmd))
            goto out;

        /* Update Config information */
        UpdateConfig((uint32_t*)(pucSrc), (uint32_t*)(pu8Response+8));
        /* Get data flash address and size */
        GetDataFlashInfo(&g_u32DataFlashAddr, &g_u32DataFlashSize);

        goto out;
    } else if(u32Lcmd == CMD_RESEND_PACKET) {
        /* for APROM & Data flash only */
        u32StartAddress -= u32LastDataLen;
        u32TotalLen += u32LastDataLen;
        if((u32StartAddress & 0xFFE00) >= CONFIG0)
            goto out;
        ReadData(u32StartAddress & 0xFFE00, u32StartAddress, (uint32_t*)s_au8ApromBuf);
        FMC_Erase_User(u32StartAddress & 0xFFE00);
        WriteData(u32StartAddress & 0xFFE00, u32StartAddress, (uint32_t*)s_au8ApromBuf);
        if((u32StartAddress%FMC_FLASH_PAGE_SIZE) >= (FMC_FLASH_PAGE_SIZE-u32LastDataLen))
            FMC_Erase_User((u32StartAddress & 0xFFE00)+FMC_FLASH_PAGE_SIZE);
        goto out;

    }

    if((u32Gcmd == CMD_UPDATE_APROM) || (u32Gcmd == CMD_UPDATE_DATAFLASH))
    {
        if(u32TotalLen < u32SrcLen) {
            /* prevent last package from over writing */
            u32SrcLen = u32TotalLen;
        }

        /* Calculate remained size */
        u32TotalLen -= u32SrcLen;

        WriteData(u32StartAddress, u32StartAddress + u32SrcLen, (uint32_t*)pucSrc);
        memset(pucSrc, 0, u32SrcLen);

        ReadData(u32StartAddress, u32StartAddress + u32SrcLen, (uint32_t*)pucSrc);
        /* Update write start address for next writing*/
```

```
        u32StartAddress += u32SrcLen;
        u32LastDataLen =  u32SrcLen;

        /* Write finished, calculate the checksum*/
        if(u32TotalLen == 0) {
            u32LckSum = CalCheckSum(u32StartAddressBak, u32TotalLenBak);
            outps(pu8Response + 8, u32LckSum);
        }
    }
out:
    u32LckSum = Checksum(pucBuffer, u8Len);
    outps(pu8Response, u32LckSum);
    ++u32GPackNo;
    outpw(pu8Response+4, u32GPackNo);
    u32GPackNo++;


    return 0;
}
```

3 軟體與硬體環境

● 軟體環境

- ■ BSP 版本
  - ◆ NUC230/240 Series BSP CMSIS v3.01.002
- ■ Android Studio 版本
  - ◆ Android studio v3.5.3
- ■ IDE 版本
  - ◆ Keil uVersion 5.28

● 硬體環境

- ■ 電路元件
  - ◆ NuTiny–EVB–NUC240–LQFP100 V1.0
  - ◆ An android device
- ■ 示意圖
  - ◆ 將 NUC240 藉由 mini USB 傳輸線和 USB OTG 傳輸線連接至 Android 裝置。

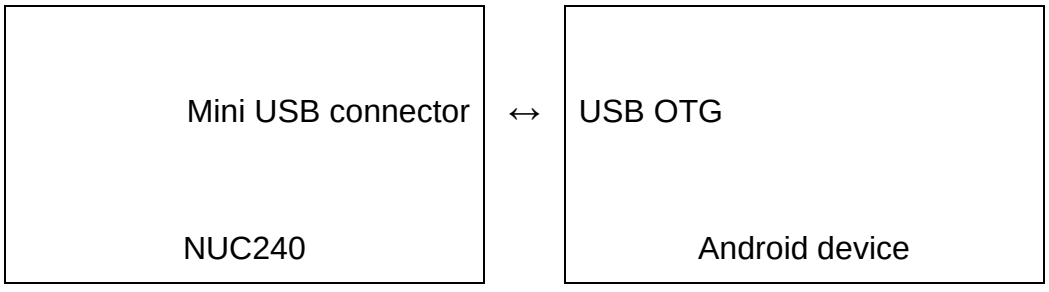

圖 3-1 NUC240 和 Android裝置連接圖

4 目錄資訊

- EC_NUC240_ANDROID_ISP_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
    - NuAndroidISPTool — Source file of android application package.

5 如何執行範例程式

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 ANDROID_ISP.uvproj。
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

6 修訂紀錄

Date	Revision	Description
Sep. 23, 2019	1.00	1. 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.