Android ISP

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	使用 NUC240 的 USB ISP 功能与 Android APP 作韧体升级
BSP 版本	NUC230/240 Series BSP v3.01.002
开发平台	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 功能介绍

1.1 简介

此范例程式演示如何在Android装置端透过USB OTG(On-The-Go)的连接来控制NUC240端的flash做韧体的升级。

1.2 原理

USB OTG是USB2.0规格的补充标准。它可使USB设备，例如播放器或手机作为主机，与USB随身碟、滑鼠或键盘连接。此范例程式使用手机(Android设备)作为主机，透过USB OTG与NUC240连接，如图1-1所示。将NuvotonISPTool应用程式安装于手机中，并以该应用程式做为控制介面，透过USB OTG来控制NUC240的Flash记忆体的擦除、写入与存取，以完成韧体的升级。

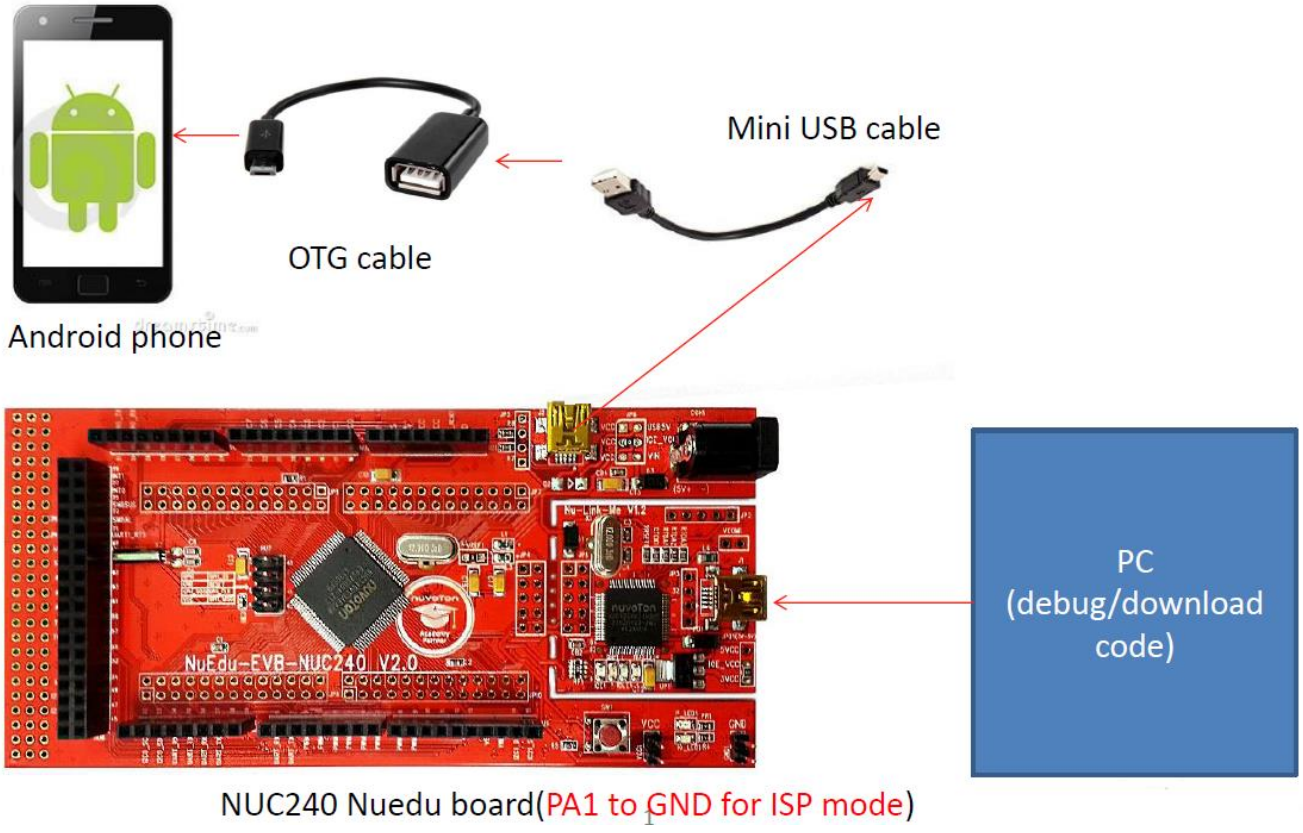

图 1-1 Android 装置与 NUC240 的连线与传输控制

1.3 执行结果

执行此范例程式前，需先在Android作业系统的手机上安装NuvotonISPTool。 NuvotonISPTool放置的目录如下：

EC_NUC240_ANDROID_ISP_V1.00\SampleCode\NuAndroidISPTool\

将该目录中的压缩档Examples_04_01.zip档解压缩后，会产生一个APK (Android Application Package)的目录如图1-2：

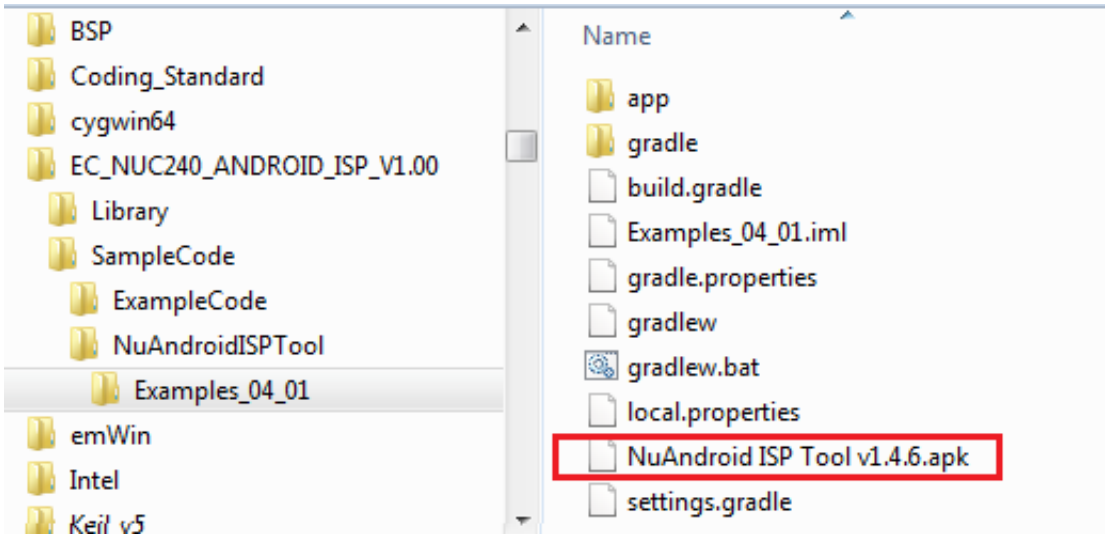

圖1-2 Android application package目录

将目录中的NuAndroid ISP Tool v1.4.6.apk下载到手机上并安装完成后，回到PC端执行范例程式。首先在PC端先开启专案ANDROID_ISP.uvproj，编译好范例程式后，将执行程式载入NUC240，并按下”Start/Stop Debug Session”功能键，如图1-3中红框所示，以进入除错模式。

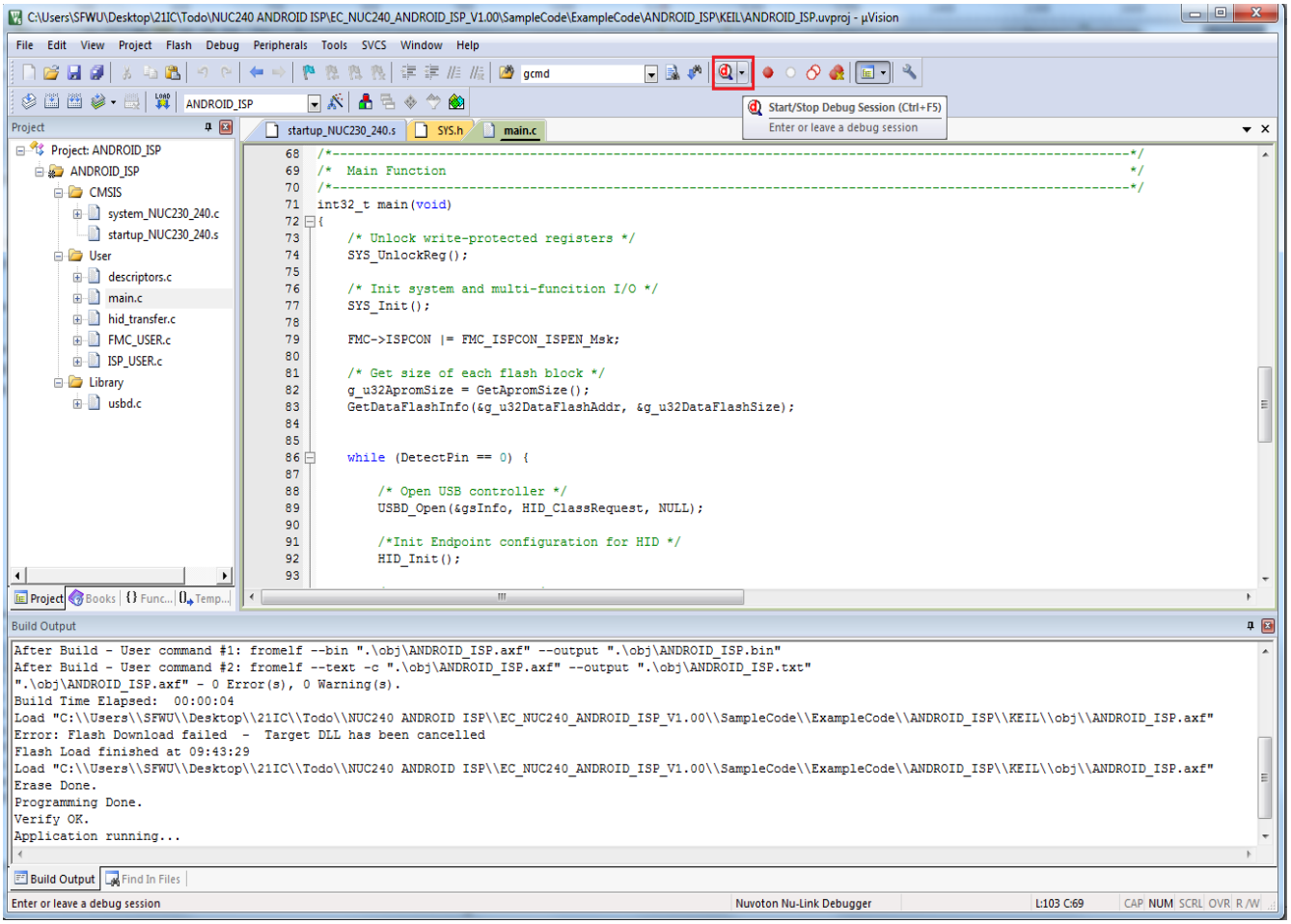

图1-3 在程式载入NUC240后，进入除错模式之功能键

进入除错模式后，按下”Run”功能键，如图1-4中红框所示，开始执行程式以启动USB连线，以等待Android 装置的连接。

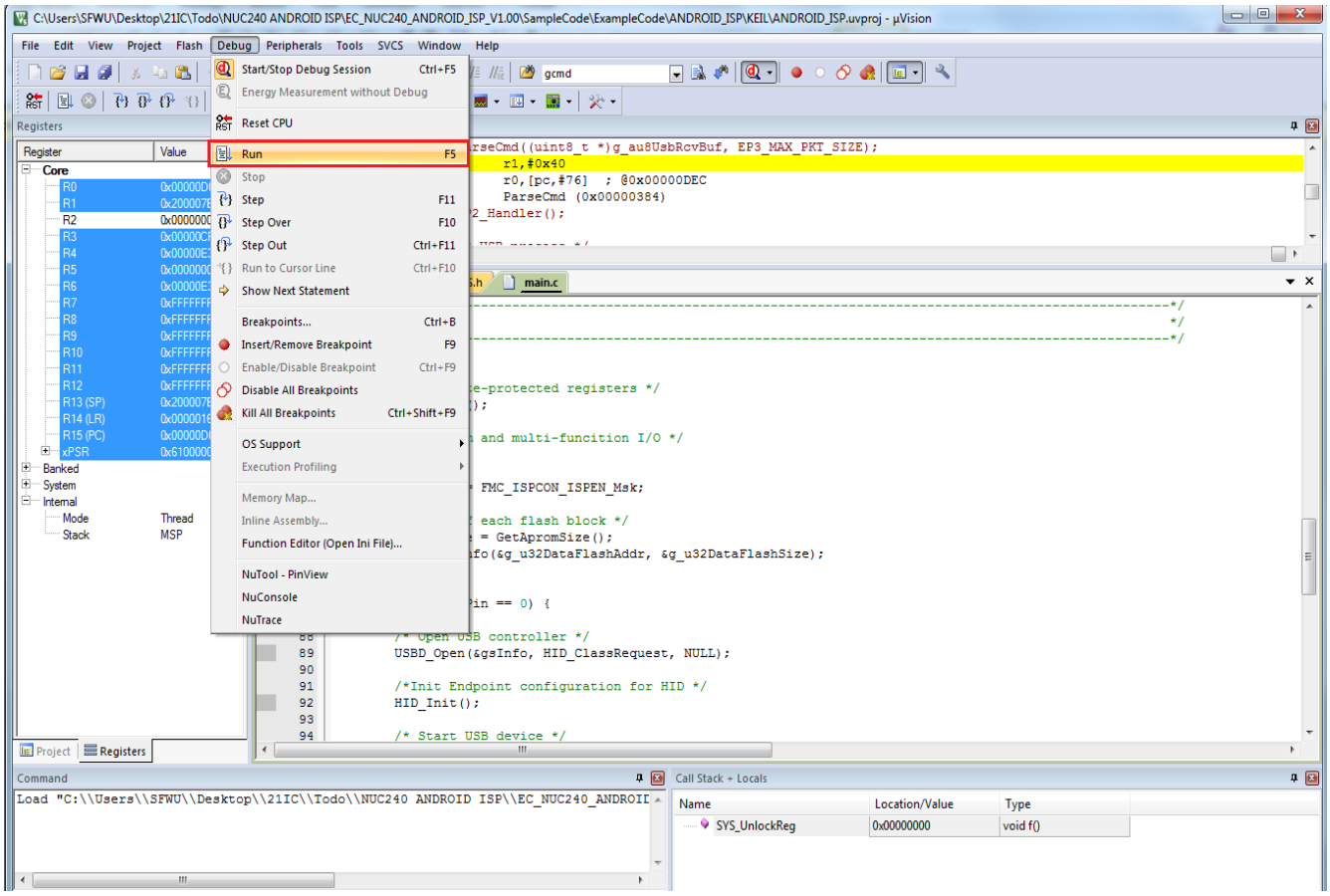

图1-4 进入除错模式，执行程式以启动USB连线

而后以OTG cable和Mini USB cable将Android手机与NUC240做连接，如章节1-2中的图1-1所示。当手机与NUC240连接完成后，在手机上会自动出现NuvotonISPTool的初始画面，并显示USB连线讯息，如图1-5所示：

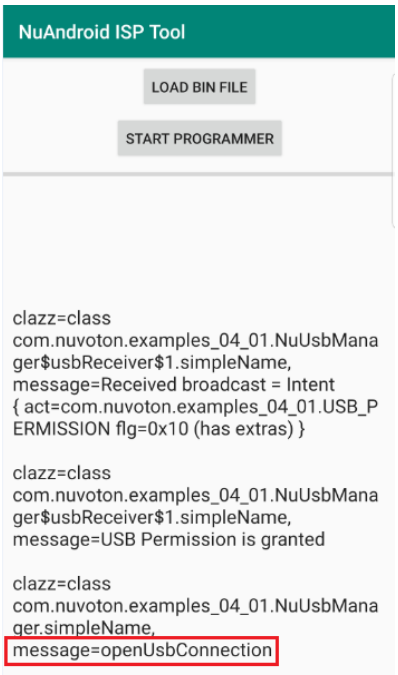

图 1-5 NuvotonISPTool 启动画面与 USB 连线讯息

NuvotonISPTool 开启后，按下”LOAD BIN FILE”功能键，以执行韧体更新，如图 1-6 所示：

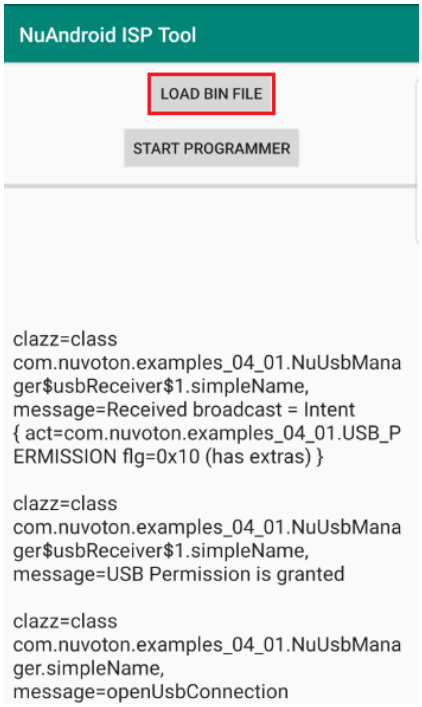

图 1-6 NuvotonISPTool 执行载入 BIN 档

按下”LOAD BIN FILE”功能键后，进入档案资料夹，选取欲写入的 BIN 档，如图 1-7 所示：

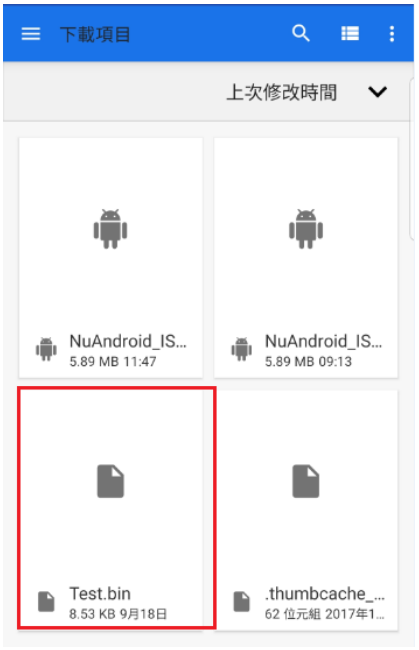

图 1-7 NuvotonISPTool 选择欲载入的 BIN 档

选择完 BIN 档之后，回到 NuvotonISPTool 的主画面，主画面换显示 BIN 档的资讯，如图 1-8 所示：

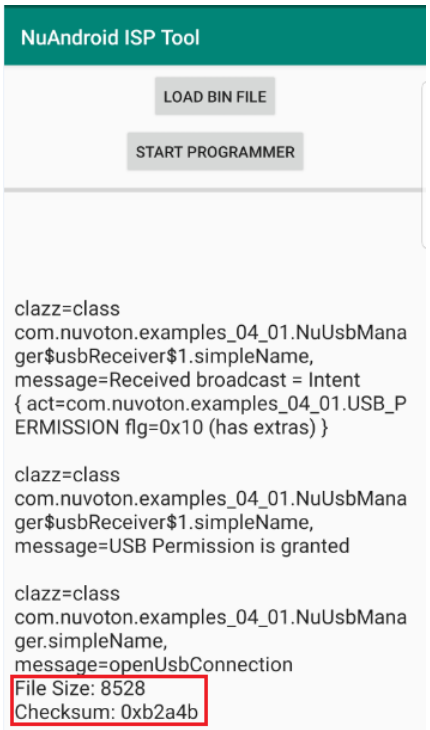

图 1-8 NuvotonISPTool 显示 BIN 档资讯

按下”START PROGRAMMER” 功能键，以将所选取的 BIN 档烧录至 NUC240 的 flash，如图 1-9 所示：

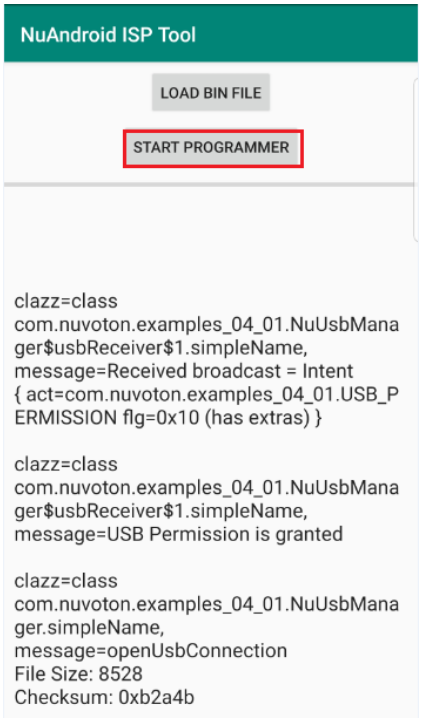

图 1-9 NuvotonISPTool 执行烧录 BIN 档

烧录完成后，NuvotonISPTool 主画面会显示烧录成功的讯息，如图 1-10 所示：

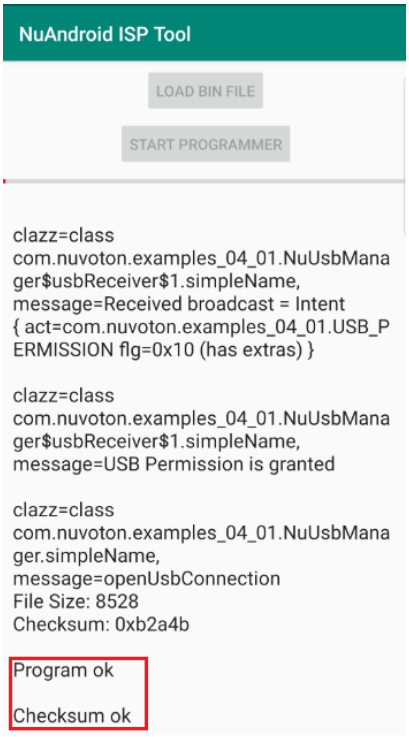

图 1-10 NuvotonISPTool 烧录成功讯息

2 代码介绍

下方的函式为主程式，主要为 USB 等待连线的流程，以及 USB 连线成功后，接收由 Android 手机端的指令以执行 Flash 的读写：

```
/*---------------------------------------------------------------------------*/
/*  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);
}
```

下方的函式为剖析 NUC240 透过 USB 接收的指令，包含 USB 连线，flash 的擦除、读写、、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 软件与硬件环境

- 软件环境
  - 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
- 硬件环境
  - 电路组件
    - NuTiny–EVB–NUC240–LQFP100 V1.0
    - An android device
  - 示意图

- 将 NUC240 藉由 mini USB 传输线和 USB OTG 传输线连接至 Android 装置。

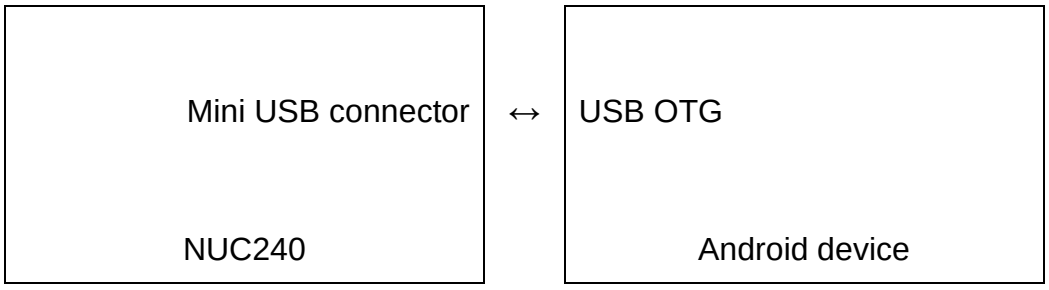

图 3-1 NUC240 和 Android装置连接图

4 目录信息

- 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 如何执行范例程序

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 ANDROID_ISP.uvproj。
2. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开除错模式
3. 进入除错模式接口
   a. 执行代码

6 修订纪录

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