Android ISP

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Use NUC240's USB ISP function and Android APP to do firmware upgrade
BSP Version	NUC230/240 Series BSP v3.01.002
Hardware	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 Function Description

1.1 Introduction

This sample code demonstrates how to upgrade the firmware of the NUC240 via the USB OTG (On-The-Go) connection on the Android device.

1.2 Principle

USB OTG is a supplement to the USB 2.0 specification. It allows a USB device, such as a player or mobile phone, to connect to a USB flash drive, mouse or keyboard. This sample code uses a mobile phone (Android device) as the host and connects to the NUC240 via USB OTG, as shown in Figure 1-1. The NuvotonISPTool application is installed in the mobile phone, and the application is used as a control interface to control the erasing, writing and reading of the NUC240's flash memory through the USB OTG connection to complete the firmware upgrade.

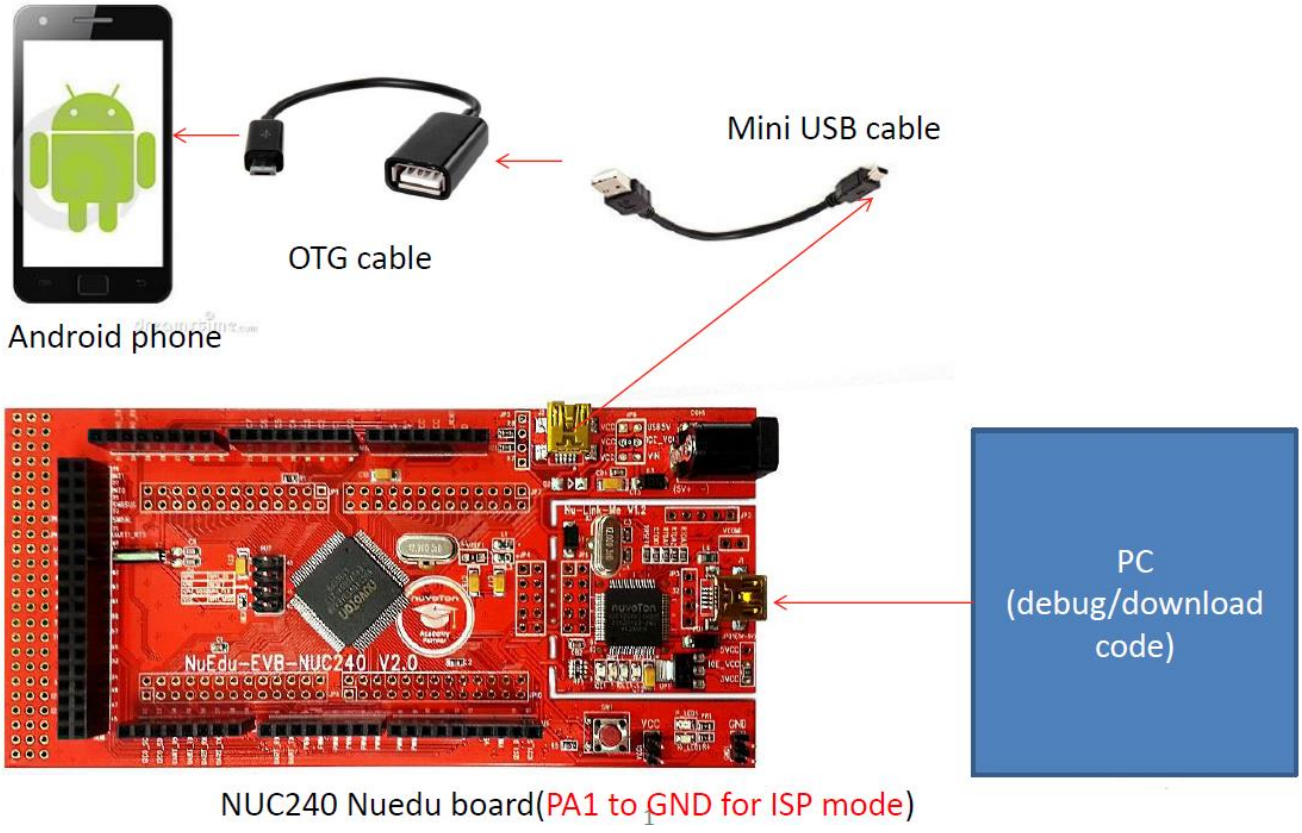

Figure 1-1 Connection and transmission control between Android device and NUC240

1.3 Demo Result

Before executing this sample code, user needs to install NuvotonISPTool on the phone with Android operating system. The path of directory of NuvotonISPTool is as follows:

EC_NUC240_ANDROID_ISP_V1.00\SampleCode\NuAndroidISPTool\

After unzipping Examples_04_01.zip file in the directory, an APK (Android Application Package) directory will be generated as shown in Figure 1-2:

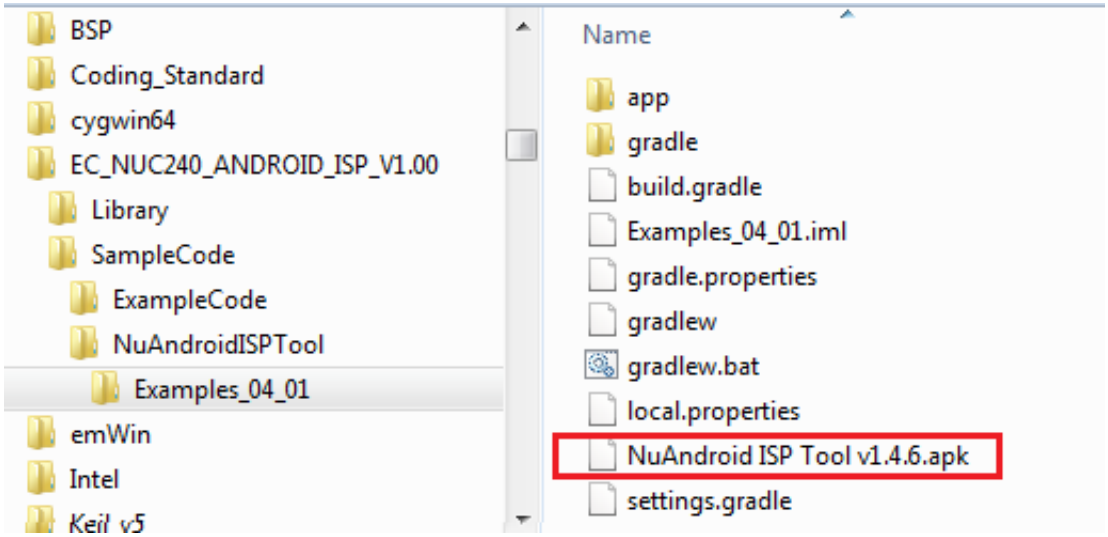

Figure 1-2 Android application package directory

Download the NuAndroid ISP Tool v1.4.6.apk from the directory to the phone and install it. Then go back to the PC and execute the sample code.

To start executing the sample code, user should open the project ANDROID_ISP.uvproj on the PC side, compile the sample program, download the program into NUC240, and click the "Start/Stop Debug Session" function button, as shown in the red box in Figure 1-3, to enter debug mode.

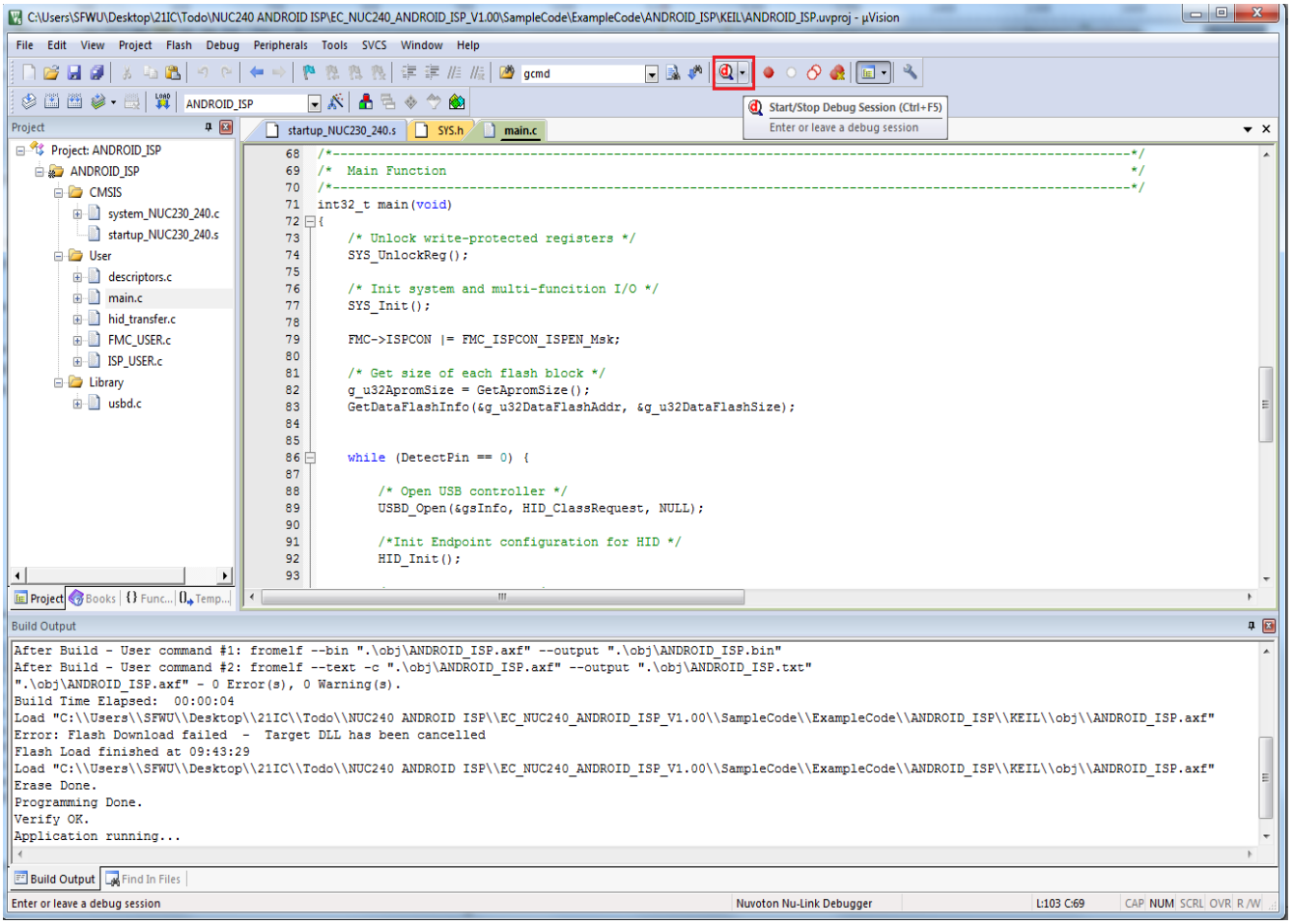

Figure 1-3 After downloading the NUC240 program, click the debug button

After entering the debug mode, click the “Run” button, as shown in the red box in Figure 1-4, start executing the program to start the USB connection to wait for the connection of the Android device.

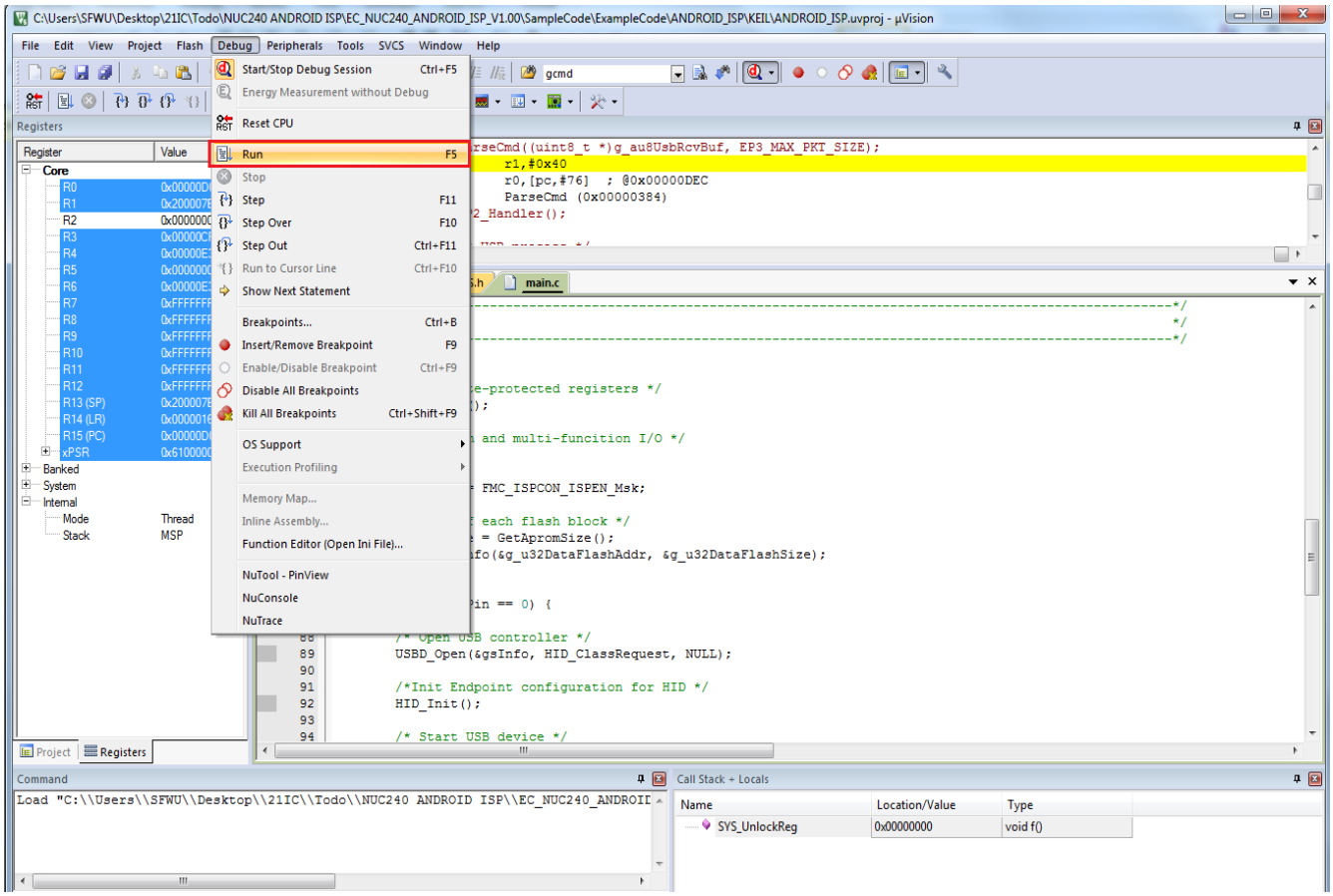

Figure 1-4 Enter debug mode, execute the program to start USB connection

Then connect the Android phone to the NUC240 with the OTG cable and the Mini USB cable, as shown in Figure 1-1 in Chapter 1-2. When the phone is connected to the NUC240, the initial screen of NuvotonISPTool will appear automatically on the phone, and the USB connection message will be displayed, as shown in Figure 1-5:

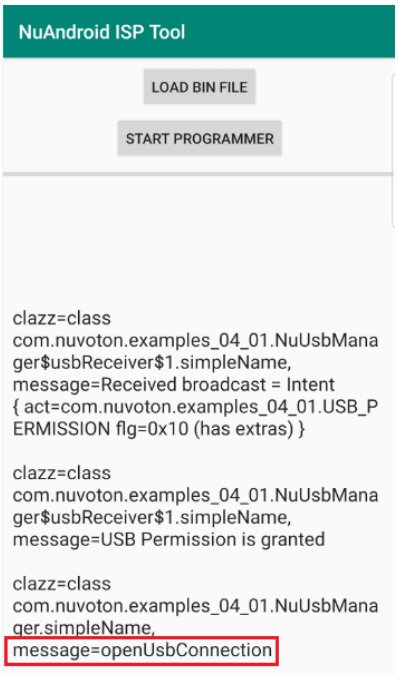

Figure 1-5 NuvotonISPTool startup screen and USB connection message

After NuvotonISPTool being turned on, click the “LOAD BIN FILE” button to perform the firmware upgrade, as shown in Figure 1-6:

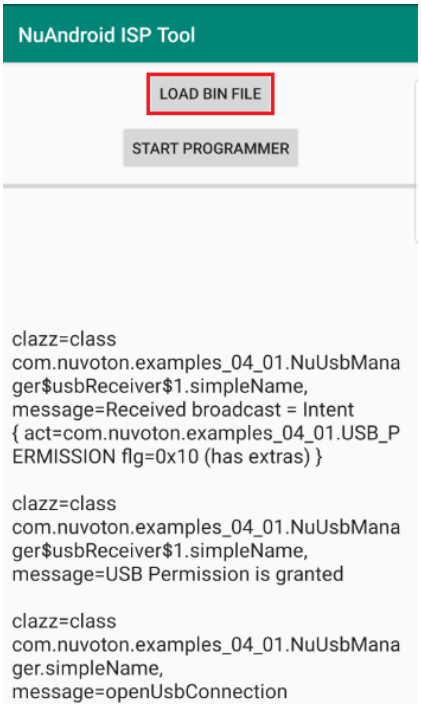

Figure 1-6 NuvotonISPTool execution of load BIN file

After clicking the "LOAD BIN FILE" function key, enter the file folder and select the BIN file to be written, as shown in Figure 1-7:

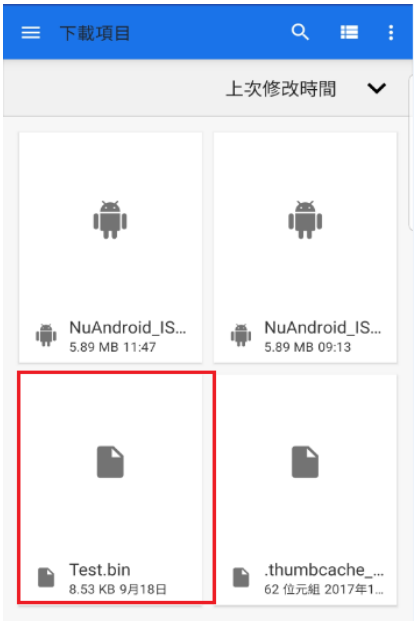

Figure 1-7 NuvotonISPTool selects BIN file to be loaded

After selecting the BIN file, return to the main screen of NuvotonISPTool, and the main screen will display the information of the BIN file, as shown in Figure 1-8:

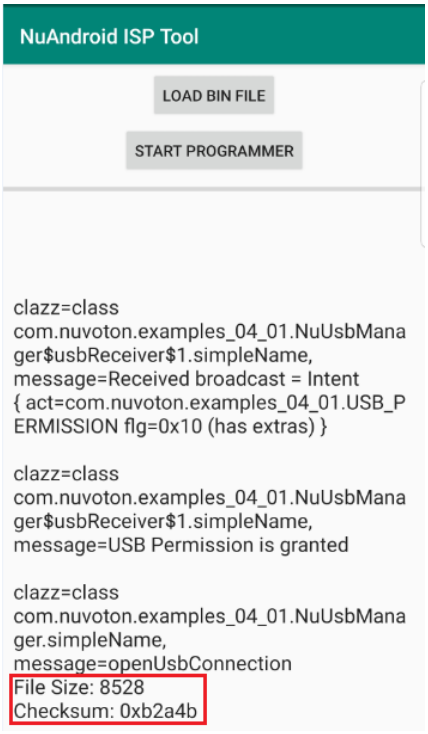

Figure 1-8 NuvotonISPTool displays BIN file information

Click the “START PROGRAMMER” button to download the selected BIN file to the flash of the NUC240, as shown in Figure 1-9:

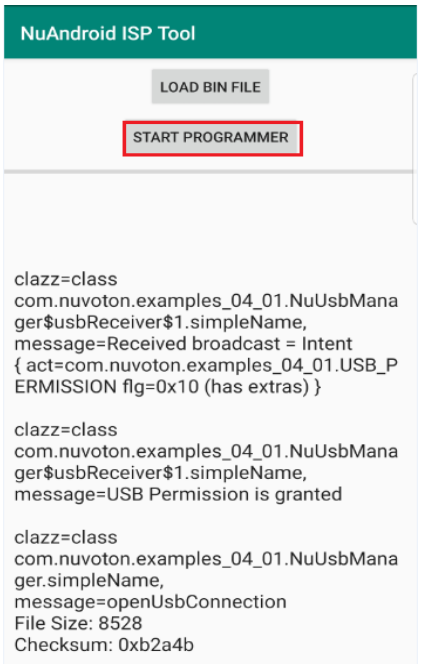

Figure 1-9 NuvotonISPTool performs downloading BIN file

After the downloading being completed, the NuvotonISPTool main screen will display the success message, as shown in Figure 1-10:

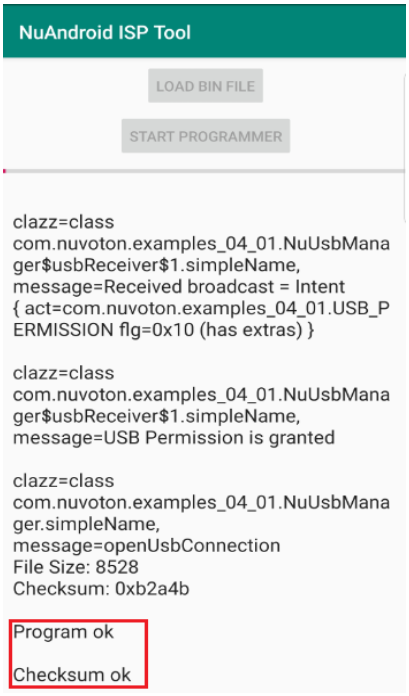

Figure 1-10 NuvotonISPTool downloading success message

2 Code Description

The function below is the main function, mainly for the process of USB waiting for connection, and after the USB connection is successful, receiving the commands from the Android mobile phone to perform flash reading and writing:

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

The function below is to parsing the commands received by the NUC240 via USB, including USB connection, flash erase, flash read, flash write, and so on:

```
/*---------------------------------------------------------------------------*/
/*  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 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP CMSIS v3.01.002
- Android studio version
  - Android studio v3.5.3
- IDE version
  - Keil uVersion 5.28

Hardware Environment

- Circuit components
  - NuTiny–EVB–NUC240–LQFP100 V1.0
  - An android device
- Diagram
  - Connect the NUC240 to the Android device via a mini USB cable and an USB OTG cable.

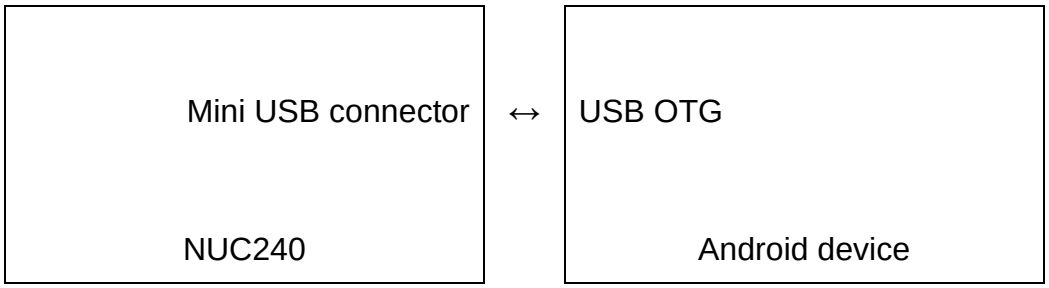

Figure 3-1 Connection diagram between NUC240 and Android device

4 Directory Information

- 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 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click ANDROID_ISP.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

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

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.