在 Linux 上驅動 Nu-Link2 存取 MCU

NuMicro® 系列微控制器範例代碼介紹

文件資訊

應用簡述	在 Linux 平台上，使用 HIDAPI 函式庫連接 Nu-Link2 並存取 NuMicro 系列微控制器。
Linux 版本	Ubuntu 20.04.3 LTS
開發平台	Nu-Link2 (Nu-Link2-Pro 或 Nu-Link2-Me)

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

展示在 Linux 上使用 HIDAPI 連接 Nu-Link2，並透過 Nu-Link2 連結目標微控制器，對目標微控制器執行晶片擦除、Flash 及配置區的寫入讀取操作。

範例中使用的 HIDAPI 函式庫，在 Linux 平台上支援了 hidraw 或 libusb 兩個後端程式(詳細請參考官網說明)。在範例編譯時，可在編譯選項裡選擇使用 hidraw 或 libusb 後端程式。

範例中支援了 8 位元(8051)和 32 位元(Arm Cortex-M)目標微控制器的存取，可在編譯選項裡選擇 8 位元或 32 位元微控制器的操作。

1.1 原理

下圖中，範例程式使用了 ICE 命令集。ICE 命令集是基於 USB HID 的協議上自定義的命令，可透過此命令集對目標微控制器做連接及控制。

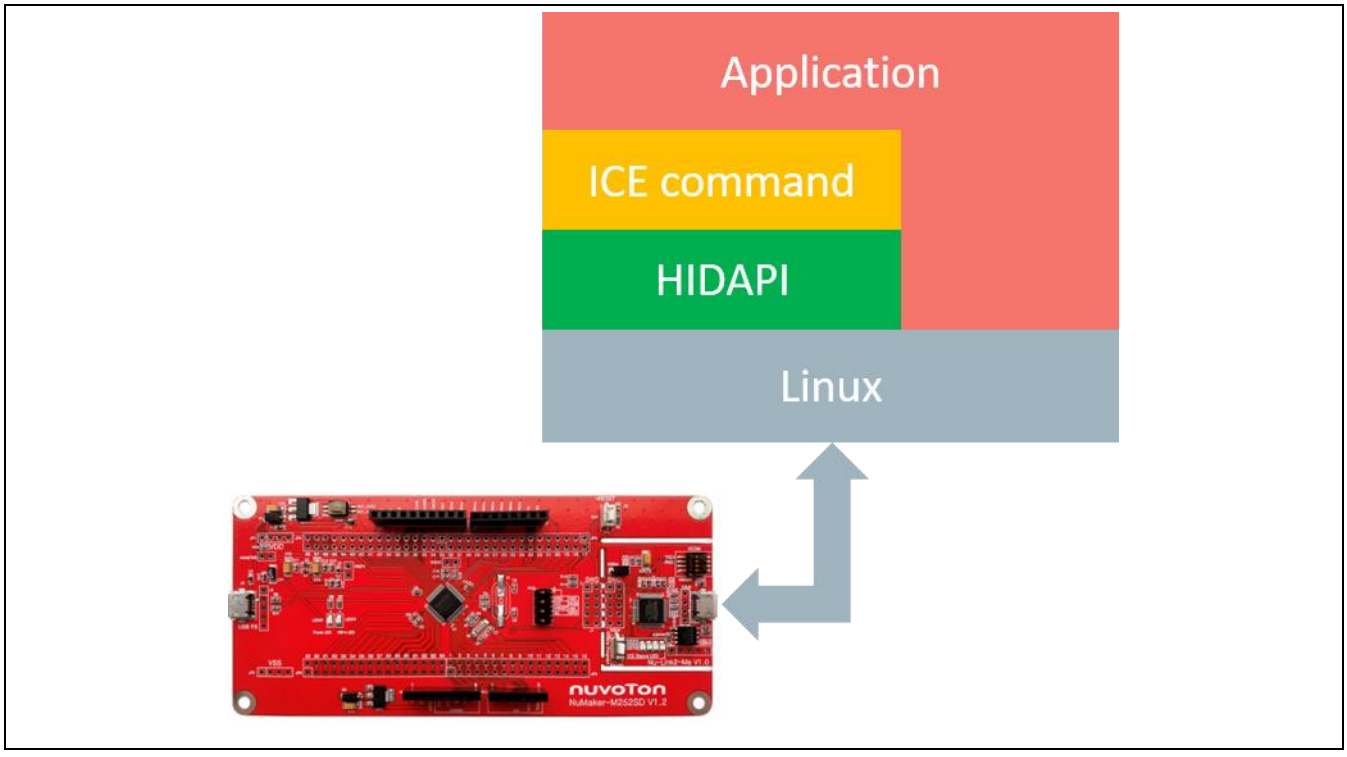

圖 1-1 架構圖

1.2 執行結果

執行後會列印出底下結果：

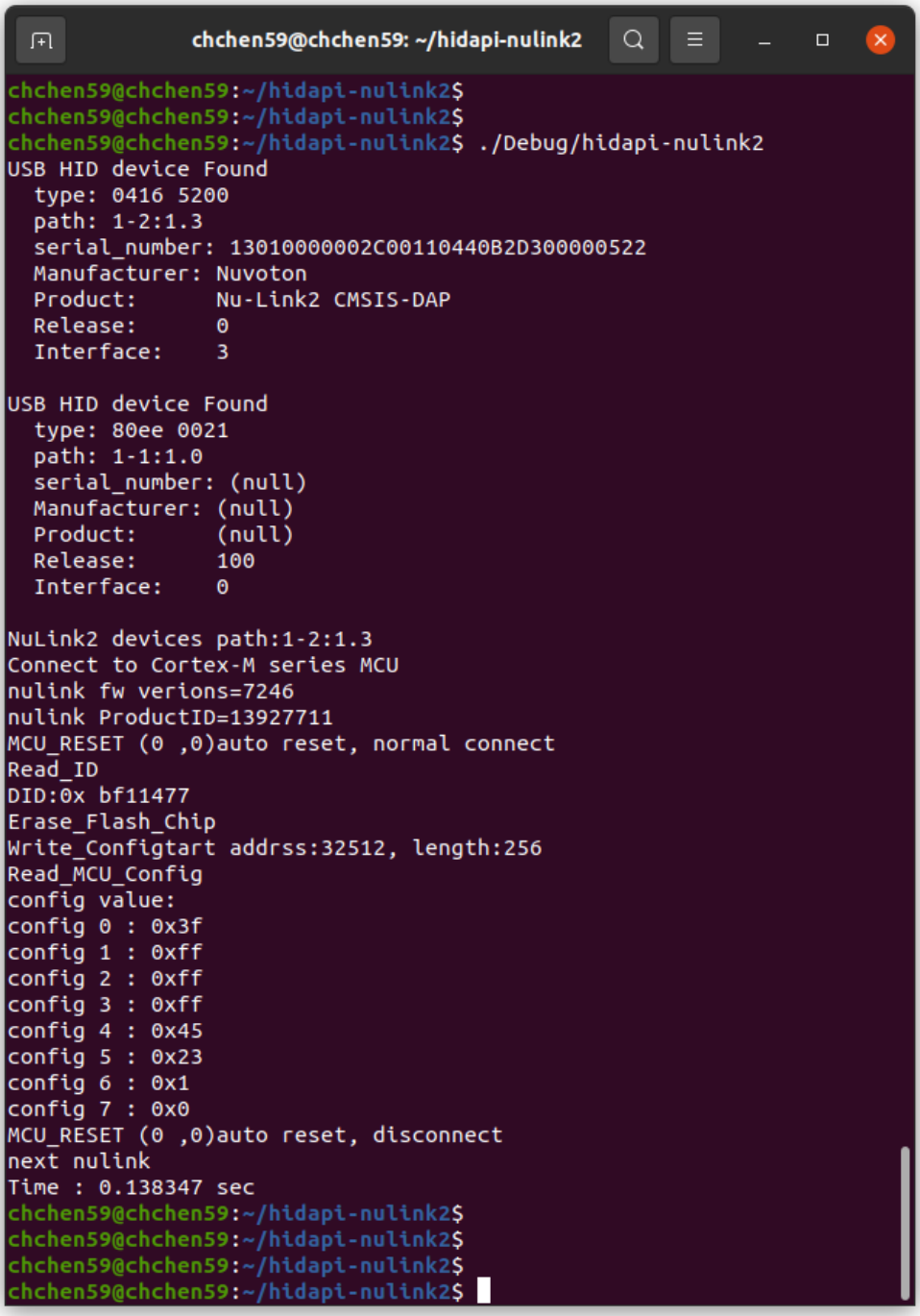

圖 1-2 執行結果

2 代碼介紹

設定 Nu-Link2 USB VID/PID 以及目標微控制器型號：

```
#define NU_LINK_USB_VID 0x0416
#define NU_LINK_USB_PID 0x5200

typedef enum
{
    eNUC_CHIP_TYPE_GENERAL_V6M= 0,
    eNUC_CHIP_TYPE_NUC1XX     = 1,
    eNUC_CHIP_TYPE_M05X       = 3,
    eNUC_CHIP_TYPE_N572       = 4,
    eNUC_CHIP_TYPE_NANO100        = 0x101,
    eNUC_CHIP_TYPE_MINI51     = 0x102,
    eNUC_CHIP_TYPE_MT5XX      = 0x103,
    eNUC_CHIP_TYPE_N570       = 0x104,
    eNUC_CHIP_TYPE_NM1120     = 0x105,
    eNUC_CHIP_TYPE_I96000     = 0x10C,
    eNUC_CHIP_TYPE_AU91XX     = 0x114,
    eNUC_CHIP_TYPE_M0564      = 0x121,
    eNUC_CHIP_TYPE_M031       = 0x10A,
    eNUC_CHIP_TYPE_M0A21      = 0x10E,
    eNUC_CHIP_TYPE_M030G      = 0x10F,
    eNUC_CHIP_TYPE_M031G      = 0x12A,
    eNUC_CHIP_TYPE_M251       = 0x201,
    eNUC_CHIP_TYPE_M2351      = 0x321,
    eNUC_CHIP_TYPE_M2354      = 0x322,
    eNUC_CHIP_TYPE_GENERAL_V7M= 0x400,
    eNUC_CHIP_TYPE_NUC4XX     = 0x401,
    eNUC_CHIP_TYPE_M451       = 0x402,
    eNUC_CHIP_TYPE_M471       = 0x40A,
    eNUC_CHIP_TYPE_M480       = 0x411,
    eNUC_CHIP_TYPE_I94000     = 0x414,
    eNUC_CHIP_TYPE_NUC505     = 0x500,
    eNUC_CHIP_TYPE_GENERAL_ARM7    = 0x700,
    eNUC_CHIP_TYPE_GENERAL_1T = 0x800, // For 8051 1T N76/ML51/MS51 Series
    eNUC_CHIP_TYPE_GENERAL_855= 0x801, // 85J, 37J
    eNUC_CHIP_TYPE_GENERAL_82J= 0x802,
    eNUC_CHIP_TYPE_GENERAL_83J= 0x803,
    eNUC_CHIP_TYPE_GENERAL_35J= 0x804,
```

```
    eNUC_CHIP_TYPE_GENERAL_87J= 0x805, // 87J, 23J
    eNUC_CHIP_TYPE_GENERAL_21J= 0x806,
    eNUC_CHIP_TYPE_GENERAL_658= 0x807,
    eNUC_CHIP_TYPE_GENERAL_659= 0x808,
    eNUC_CHIP_TYPE_GENERAL_ARM9    = 0x900
}E_NUC_CHIP_TYPE;

//Target chip type
#if (TARGET == TARGET_CORTEX_M)
E_NUC_CHIP_TYPE s_eNUCChipType = eNUC_CHIP_TYPE_M251;
#else
E_NUC_CHIP_TYPE s_eNUCChipType = eNUC_CHIP_TYPE_GENERAL_1T;
#endif
```

USB HID 裝置列舉並找出 Nu-Link2 裝置節點：

```
    //enum nulink 2, to compare vendor_id == 0x0416, and product_id==0x5200 and record
devices path
    char dest_path[MAX_NULINK2_COUNT][64];
    struct hid_device_info *devs, *cur_dev;
    devs = hid_enumerate(0x0, 0x0);
    cur_dev = devs;
    int device_count = 0;
    while (cur_dev) {
        printf("Device Found\n  type: %04hx %04hx\n  path: %s\n  serial_number: %ls",
            cur_dev->vendor_id, cur_dev->product_id, cur_dev->path, cur_dev-
>serial_number);
        printf("\n");
        printf("  Manufacturer: %ls\n", cur_dev->manufacturer_string);
        printf("  Product:      %ls\n", cur_dev->product_string);
        printf("  Release:      %hx\n", cur_dev->release_number);
        printf("  Interface:    %d\n", cur_dev->interface_number);
        printf("\n");



        if ((cur_dev->vendor_id == NU_LINK_USB_VID)&&(cur_dev->product_id ==
NU_LINK_USB_PID))
        {

            strcpy(dest_path[device_count], cur_dev->path);  //copy path name to buffer.
            device_count++;
        }
        cur_dev = cur_dev->next;
```

```
    }
    hid_free_enumeration(devs);
```

目標微控制器連接：

```
int connect_target_chip(void)
{
#if (TARGET == TARGET_CORTEX_M)
    //command flow
    //Reset command serial
    if (ResetCmdSerial() != 0)
        return -1;
    //set config nulink
    if (Set_Config_NULINK() != 0)
        return -2;
    //mode entry for 32bit mcu
    if (MCU_RESET_CONNECT_32() != 0)
        return -3;

    //Read target chip DID
    if (Read_ID_32() != 0) //READ device id
        return -4;

#else
    //Reset command serial
    if (ResetCmdSerial() != 0)
        return -1;
    //set config nulink
    if (Set_Config_NULINK() != 0)
        return -2;
    //mode entry for 8bit mcu
    if (Mode_Entry() != 0)
        return -3;
    //entry icp mode
    if (ICP_Mode_Entry() != 0)
        return -4;
    //Read target chip UID, DID
    if (Read_ID() != 0) //READ device id
        return -5;

#endif
```

```
    return 0;
}
```

晶片擦除：

```
int Erase_Flash_Chip(void)
{
    printf("Erase_Flash_Chip\n\r");
    int res;
    int false_flag = 0;
    unsigned int local_retry = 0;
    memset(USBBUF_OUT, 0x0, NULINK_PACKAGE_LENGTH + 1); //clear out buffer and report id
    memset(USBBUF_IN, 0X0, NULINK_PACKAGE_LENGTH);//clear in buffer
    USBBUF_OUT[1] = s_u8Index;
    set_out_length(2 * 4);
    outpw(&USBBUF_OUT[4], 0xA4); //command
    outpw(&USBBUF_OUT[4 + 4], 0X00);//Reset_Options:

    USBBUF_OUT[0] = 0;//usb report id, always is 0
    res = hid_write(s_pHandle, USBBUF_OUT, 513);
    while (1)
    {
        res = hid_read(s_pHandle, USBBUF_IN, 512);
        if (USBBUF_IN[0] == s_u8Index&&(res == 512))
            break;
        if (local_retry > USB_TRY_CNT)
        {
            false_flag = 1;
            return false_flag;
        }
    }

    unsigned int return_command;
    return_command = inpw(&USBBUF_IN[3]);
    dbgprint("nulink return_command=0x%x\n\r", return_command);
    index_inc();
    return false_flag;
}
```

Flash 存取 (須先執行晶片擦除動作)：

```
int flash_access(void)
{
    //clear read memory buffer, and set write buffer
    //for testing, flash content is a fixed pattern
    //you can put your flash content to s_au8MemFlashArray
    for (int i = 0; i < MEMORY_SIZE_TEST; i++)
    {
        s_au8MemFlashArrayRead[i] = 0;
        s_au8MemFlashArray[i] = i & 0xff;
    }
    //program flash
    for (unsigned int i = 0; i < MEMORY_SIZE_TEST; i = i + 32)
    {
        Write_Flash(i, PAGE_PROGRAM_SIZE, &s_au8MemFlashArray[i]);

    }
    //read flash context
    for (unsigned int i = 0; i < MEMORY_SIZE_TEST; i = i + 256)
    {
        Read_Flash(i, READ_SIZE, &s_au8MemFlashArrayRead[i]);
    }
    //compare memory
    for (int i = 0; i < MEMORY_SIZE_TEST; i++)
    {
        if (s_au8MemFlashArrayRead[i] != s_au8MemFlashArray[i])
        {
            printf("error =%d\n\r", i);
            break;
        }
    }

    return 0;
}
```

配置區存取：

```
int config_access(void)
{
    //set config buffer

#if (TARGET == TARGET_CORTEX_M)
    //config 0, 32bit, Low byte first
```

```
    s_au8ConfigArray[0] = 0x3f; //see trm doc for config value setting
    s_au8ConfigArray[1] = 0xff;
    s_au8ConfigArray[2] = 0xff;
    s_au8ConfigArray[3] = 0xff;

    //config 1, 32bit, Low byte first
    s_au8ConfigArray[4] = 0x45; //see trm doc for config value setting
    s_au8ConfigArray[5] = 0x23;
    s_au8ConfigArray[6] = 0x01;
    s_au8ConfigArray[7] = 0x00;

    Write_Config_32(s_au8ConfigArray);

    //read config from buffer
    Read_MCU_Config_32(s_au8ConfigArrayRead);
    printf("config value:\n\r");
    for (int i = 0; i < 8; i++)
    {
        printf("config %d : 0x%x\n\r", i, s_au8ConfigArrayRead[i]);
        if (s_au8ConfigArrayRead[i] != s_au8ConfigArray[i])
            printf("error =%d\n\r", i);
    }

#else
    s_au8ConfigArray[0] = 0x6f; //see trm doc for config value setting
    s_au8ConfigArray[1] = 0xff;
    s_au8ConfigArray[2] = 0xff;
    s_au8ConfigArray[3] = 0xff;
    s_au8ConfigArray[4] = 0xff;
    Write_Config(s_au8ConfigArray);

    //read config from buffer
    Read_MCU_Config(s_au8ConfigArrayRead);
    printf("config value:\n\r");
    for (int i = 0; i < 5; i++)
    {
        printf("config %d : 0x%x\n\r", i, s_au8ConfigArrayRead[i]);
        if (s_au8ConfigArrayRead[i] != s_au8ConfigArray[i])
            printf("error =%d\n\r", i);
    }
```

```
#endif

    return 0;
}
```

3 軟體與硬體需求

3.1 軟體需求

- Linux 版本
  - Ubuntu 20.04.3 LTS
- HIDAPI
  - https://github.com/libusb/hidapi

3.2 硬體需求

- Nu-Link2 (Nu-Link2-Pro 或 Nu-Link2-Me)
- 微控制器開發板 (例如: NuMaker-IoT-M252, NuMaker-ML56SD)

4 目錄資訊

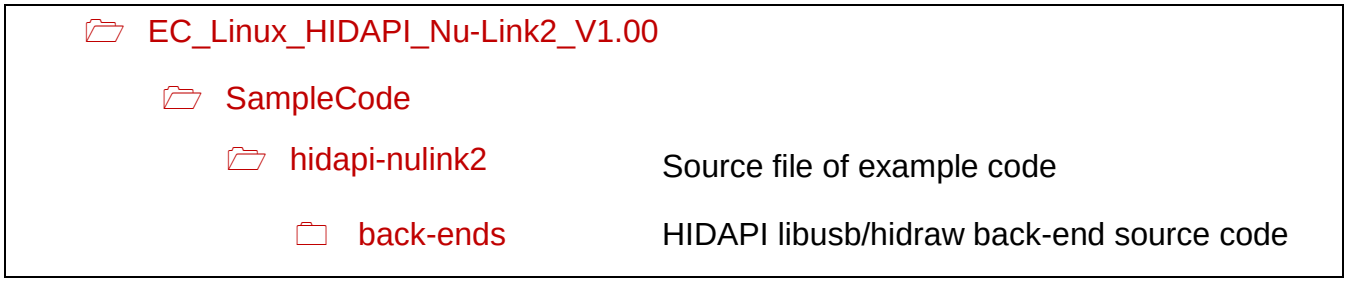

圖 4-1 目錄資訊

5 範例程式執行

1. 安裝相關工具及函式庫

   - 編譯工具
     - #sudo apt install autotools-dev autoconf automake libtool
   - USB 相關函式庫
     - #sudo apt install libudev-dev libusb-1.0-0-dev

2. 由於會使用 Nu-Link2 ICE 命令的寫入功能，因此需要開啟 Nu-Link2 HID 裝置節點寫入權限。寫入權限的開啟則是使用 udev 做動態裝置管理，因此需要為 Nu-Link2 新增 udev 規則。在 hidapi-nulink2 的資料夾中有一預設的 udev 規則文件(nulink2.rules)，需確認文件裡的 idVendor 與 idProduct 設定和 Nu-Link2 的 USB VID/PID 是否一致，最後複製此份規則文件到 /etc/udev/rules.d 目錄下。

   - 連接 Nu-Link2 後，使用 lsusb 查詢 Nu-Link2 的資訊。
   - 打開規則文件，確認 VID/PID 一致性。

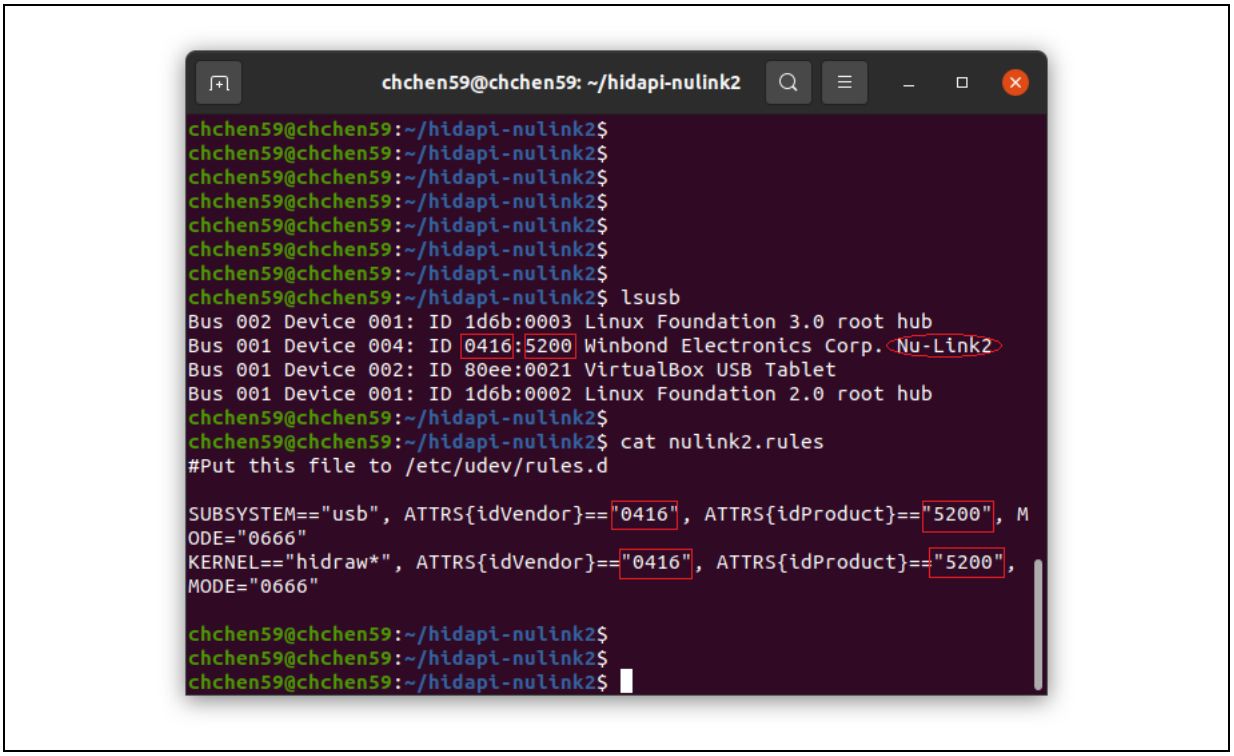

圖 5-1Nu-Link2 規則文件

3. 根據目錄資訊章節進入 hidapi-nulink2 資料夾，執行編譯腳本。

   - 編譯選項
     - -b libusb/hidraw: 選擇 HIDAPI 後端程式，預設 libusb。
     - -t 8051/cortex-m: 選擇目標微控制器種類，預設 cortex-m。
   - 編譯
     - #./build.sh –b libusb –t cortex-m

4. 連接 Nu-Link2 和目標微控器開發板後，在 Debug 目錄下執行範例程式。

   - #./hidapi-nulink2

6 修訂紀錄

Date	Revision	Description
2021.11.04	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.