在 Linux 上通过 Nu-Link2 读写 MCU

NuMicro® 系列微控制器范例代码介绍

文件信息

应用简述	在 Linux 平台上，使用 HIDAPI 函数库连接 Nu-Link2 并读写 NuMicro 系列微控制器。
Linux 版本	Ubuntu 20.04.3 LTS
开发平台	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

本范例展示在 Linux 上使用 HIDAPI 连接 Nu-Link2，并通过 Nu-Link2 连接目标微控制器，对目标微控制器执行芯片擦除、Flash 及配置区的写入读取操作。

范例中使用的 HIDAPI 函数库，在 Linux 平台上支持了 hidraw 或 libusb 两个后端程序(详细请参考官网说明)。在范例编译时，可在编译选项里选择使用 hidraw 或 libusb 后端程序。

范例中支持了 8 位(8051)和 32 位(Arm Cortex-M)目标微控制器的存取，可在编译选项里选择 8 位或 32 位微控制器的操作。

1.1 原理

下图中，范例程序使用了 ICE 命令集。ICE 命令集是基于 USB HID 的协议上自定义的命令，可通过此命令集对目标微控制器做连接及控制。

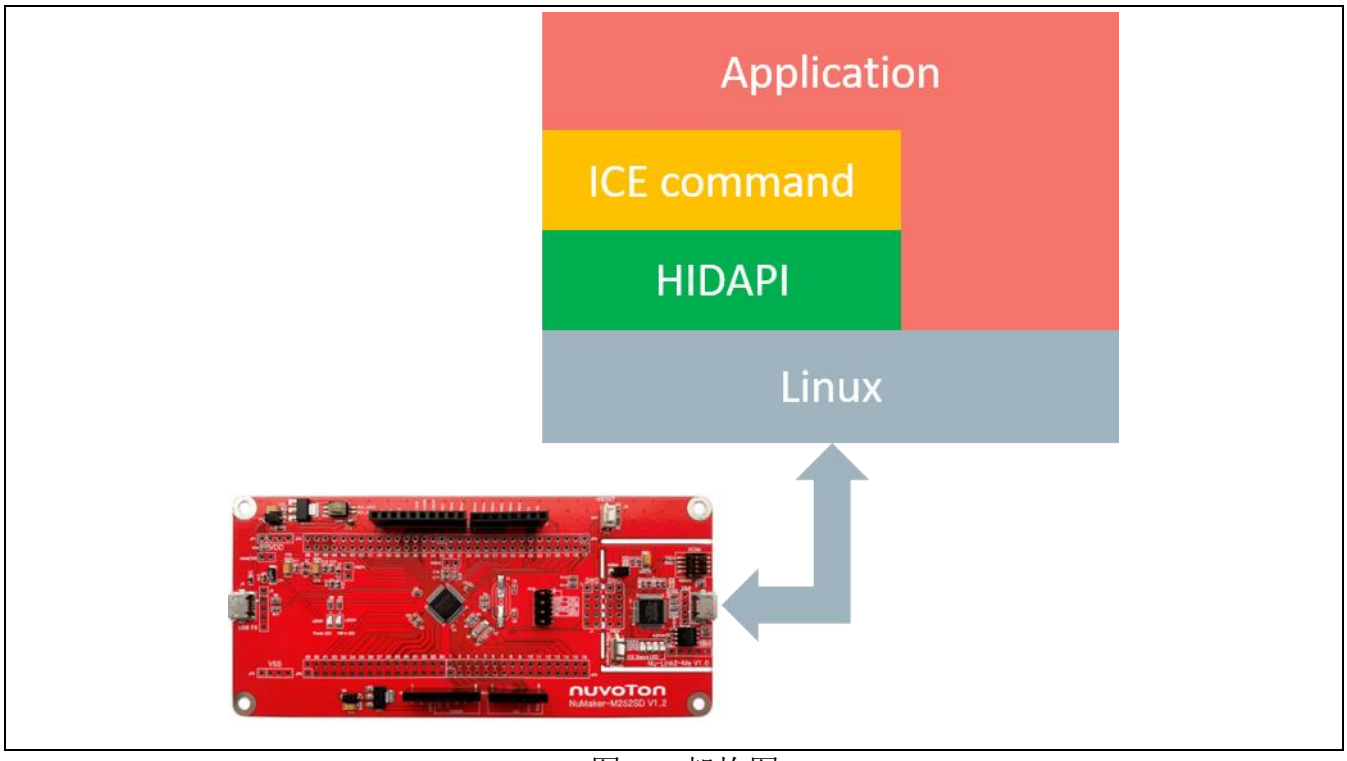

图 1-1 架构图

1.2 执行结果

执行结果，以及打印数据。

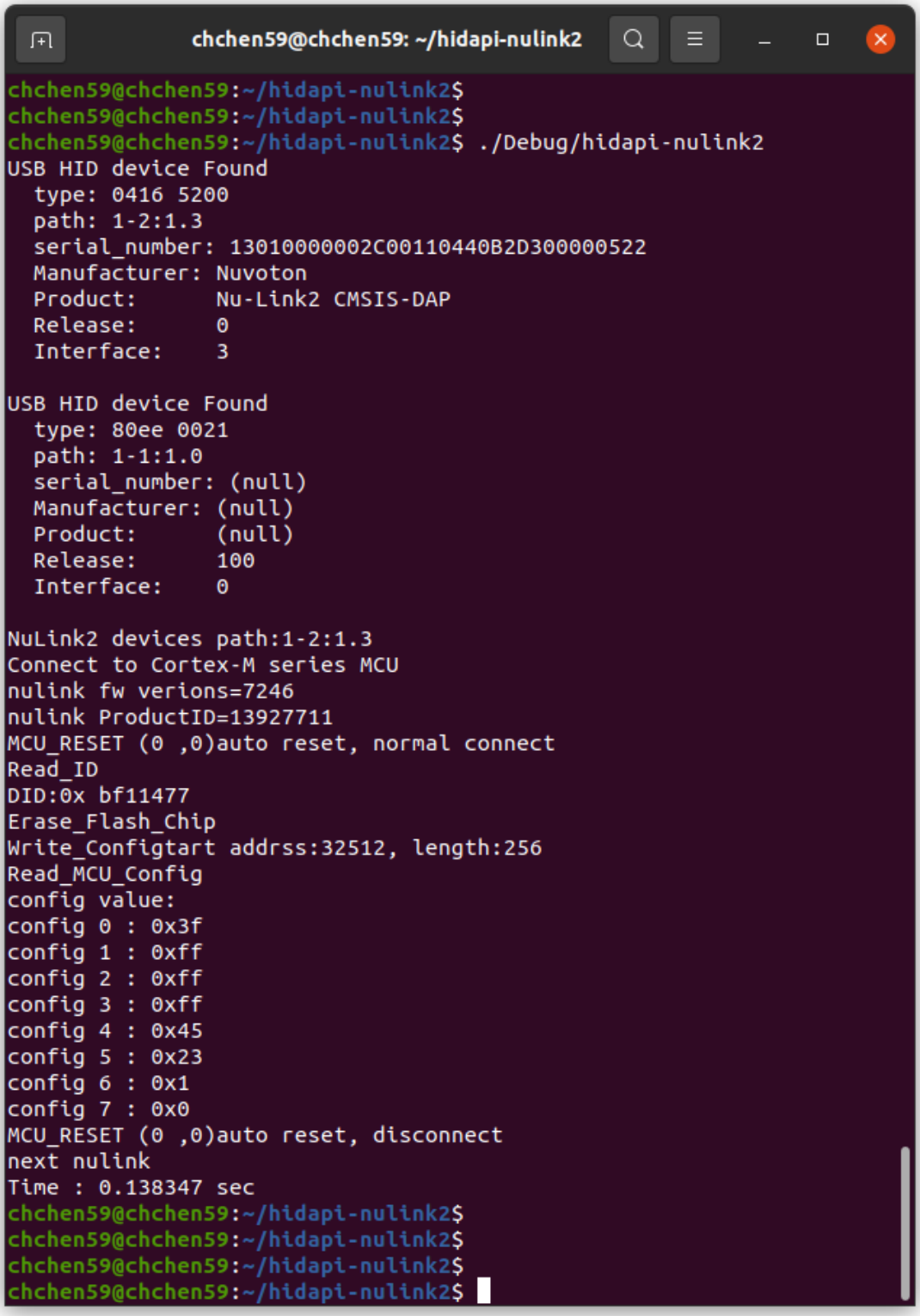
chchen59@chchen59: ~/hidapi-nulink2
chchen59@chchen59:~/hidapi-nulink2$
chchen59@chchen59:~/hidapi-nulink2$
chchen59@chchen59:~/hidapi-nulink2$ ./Debug/hidapi-nulink2
USB HID device Found
type: 0416 5200
path: 1-2:1.3
serial_number: 13010000002C00110440B2D300000522
Manufacturer: Nuvoton
Product: Nu-Link2 CMSIS-DAP
Release: 0
Interface: 3
USB HID device Found
type: 80ee 0021
path: 1-1:1.0
serial_number: (null)
Manufacturer: (null)
Product: (null)
Release: 100
Interface: 0
NuLink2 devices path:1-2:1.3
Connect to Cortex-M series MCU
nulink fw verions=7246
nulink ProductID=13927711
MCU_RESET (0 ,0)auto reset, normal connect
Read_ID
DID:0x bf11477
Erase_Flash_Chip
Write_Configtart addrss:32512, length:256
Read_MCU_Config
config value:
config 0 : 0x3f
config 1 : 0xff
config 2 : 0xff
config 3 : 0xff
config 4 : 0x45
config 5 : 0x23
config 6 : 0x1
config 7 : 0x0
MCU_RESET (0 ,0)auto reset, disconnect
next nulink
Time : 0.138347 sec
chchen59@chchen59:~/hidapi-nulink2$
chchen59@chchen59:~/hidapi-nulink2$
chchen59@chchen59:~/hidapi-nulink2$
chchen59@chchen59:~/hidapi-nulink2$

图 1-2 执行结果

2 代码介绍

设定 Nu-Link2 USB VID/PID 以及目标微控制器型号：

```
#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 装置列举并找出 Nu-Link2 设备节点：

```
    //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);
```

目标微控制器连接：

```
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;
}
```

芯片擦除：

```
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 (须先执行芯片擦除动作)：

```
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;
}
```

配置区存取：

```
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 软件与硬件需求

3.1 软件需求

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

3.2 硬件需求

- Nu-Link2 (Nu-Link2-Pro 或 Nu-Link2-Me)
- 微控制器开发板 (例如: NuMaker-IoT-M252, NuMaker-ML56SD)

4 目录信息

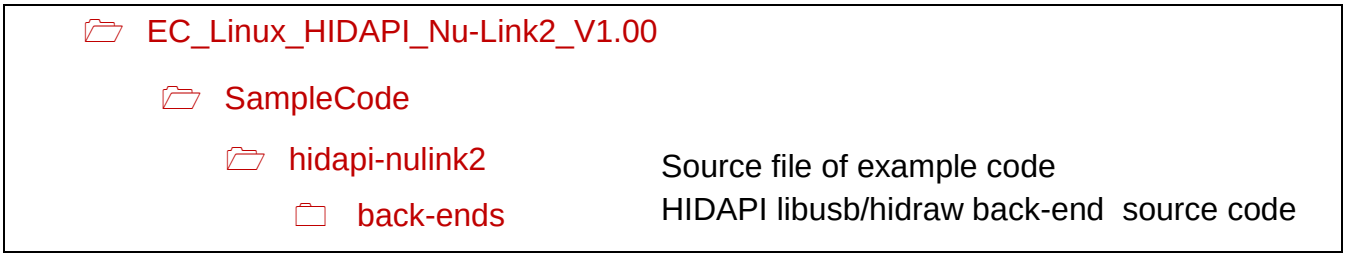

图 4-1 目录信息

5 范例程序执行

1. 安装相关工具及函数库
   - 编译工具
     - #sudo apt install autotools-dev autoconf automake libtool
   - USB 相关函数库
     - #sudo apt install libudev-dev libusb-1.0-0-dev
2. 由于会使用 Nu-Link2 ICE 命令的写入功能，因此需要开启 Nu-Link2 HID 设备节点写入权限。写入权限的开启则是使用 udev 做动态设备管理，因此需要为 Nu-Link2 新增 udev 规则。在 hidapi-nulink2 的文件夹中有一默认的 udev 规则文件(nulink2.rules)，需确认文件里的 idVendor 与 idProduct 设定和 Nu-Link2 的 USB VID/PID 是否一致，最后复制此份规则文件到 /etc/udev/rules.d 目录下。
   - 连接 Nu-Link2 后，使用 lsusb 查询 Nu-Link2 的信息。
   - 打开规则文件，确认 VID/PID 一致性。

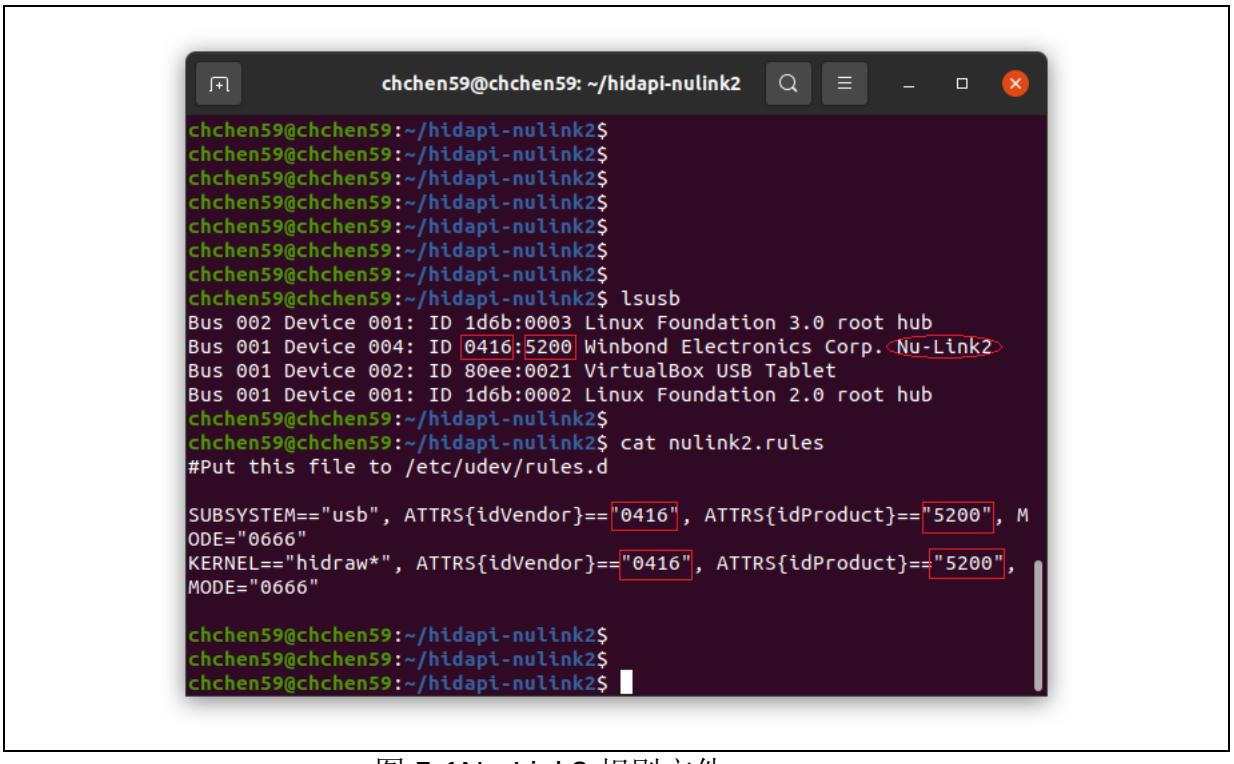

图 5-1Nu-Link2 规则文件

3. 根据目录信息章节进入 hidapi-nulink2 文件夹，执行编译脚本。
   - 编译选项

- -b libusb/hidraw: 选择 HIDAPI 后端程序，默认 libusb。
- -t 8051/cortex-m: 选择目标微控制器种类，预设 cortex-m。

■ 编译

- #./build.sh –b libusb –t cortex-m

4. 连接 Nu-Link2 和目标微控器开发板后，在 Debug 目录下执行范例程序。

■ #./hidapi-nulink2

6 修订纪录

Date	Revision	Description
2021.11.09	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.