云天励飞DESDK设备管理工具使用手册

版本：v1.0.0

更新记录

版本	修改日期	修改说明
V1.0.0	2020.11.24	初始版本

目录

1 概述

DESDK设备管理工具是一套集设备信息查询、固件下载、复位控制、文件传输、设备测试功能为一体的测试工具，随DESDK版本一起发布，提供多个平台和系统支持。

2 功能说明

2.1 帮助

```
# ./desdk -h
Usage:
-h                      : display this help
-v                      : display tool version
info                    : dump device info, add -h to get more info
set                     : config device info, -h to get more info
download                : burn device firmware, add -h to get more info
transfer                : transfer file/command from [to] device, add -h to get
more info
test                    : stress test the device, add -h to get more info
```

2.2 获取工具版本号

```
# ./desdk -v
version: v1.0
```

2.3 获取信息

2.3.1 获取usb信息

2.3.1.1 不导入环境变量

```
# ./desdk info -t usb -l
```

```
Warning!!! Please export USB_CONFIG_INI to get more info
x: abnormal     -: not support
-------------------------------------------------------------------------------------------------------------
|card_id   |index   |dev_id   |key         |ttyACMx   |slot_name      |speed   |busnum   |devnum   |mode   |status   |
-------------------------------------------------------------------------------------------------------------
|x         |x       |x        |0x1c00001   |ttyACM1   |0000:00:1c.0   |50000   |4        |46       |-      |kernel   |
|x         |x       |x        |0x1c00002   |ttyACM2   |0000:00:1c.0   |50000   |4        |47       |-      |kernel   |
|x         |x       |x        |0x1c00003   |ttyACM3   |0000:00:1c.0   |50000   |4        |48       |-      |kernel   |
|x         |x       |x        |0x1c00004   |ttyACM0   |0000:00:1c.0   |50000   |4        |45       |-      |kernel   |
-------------------------------------------------------------------------------------------------------------
```

注：如果工具没有找到USB_CONFIG_INI环境变量，则card_id/index/dev_id 将无法打印出来

2.3.1.2 导入环境变量

```
# export USB_CONFIG_INI=/root/usbprop.ini
# ./desdk info -t usb  -l
```

```
x: abnormal     -: not support
-------------------------------------------------------------------------------------------------------------
|card_id   |index   |dev_id   |key         |ttyACMx   |slot_name      |speed   |busnum   |devnum   |mode   |status   |
-------------------------------------------------------------------------------------------------------------
|0         |0       |0        |0x1c00001   |ttyACM1   |0000:00:1c.0   |50000   |4        |46       |-      |kernel   |
|0         |1       |1        |0x1c00002   |ttyACM2   |0000:00:1c.0   |50000   |4        |47       |-      |kernel   |
|0         |2       |2        |0x1c00003   |ttyACM3   |0000:00:1c.0   |50000   |4        |48       |-      |kernel   |
|0         |3       |3        |0x1c00004   |ttyACM0   |0000:00:1c.0   |50000   |4        |45       |-      |kernel   |
-------------------------------------------------------------------------------------------------------------
```

2.3.1.3 获取usb信息以及usb模式

```
# export USB_CONFIG_INI=/root/usbprop.ini
# ./desdk info -t usb  -l -m
```

```
x: abnormal     -: not support
-------------------------------------------------------------------------------------------------------------
|card_id   |index   |dev_id   |key         |ttyACMx   |slot_name      |speed   |busnum   |devnum   |mode   |status   |
-------------------------------------------------------------------------------------------------------------
|0         |0       |0        |0x1c00001   |ttyACM3   |0000:00:1c.0   |50000   |4        |40       |2      |kernel   |
|0         |1       |1        |0x1c00002   |ttyACM0   |0000:00:1c.0   |50000   |4        |37       |2      |kernel   |
|0         |2       |2        |0x1c00003   |ttyACM1   |0000:00:1c.0   |50000   |4        |38       |2      |kernel   |
|0         |3       |3        |0x1c00004   |ttyACM2   |0000:00:1c.0   |50000   |4        |39       |2      |kernel   |
-------------------------------------------------------------------------------------------------------------
```

注：由于usb模式的获取可能比较耗时，或者部分设备不支持usb模式，所以单独添加-m

2.3.1.4 usb信息说明

card_id: 只在pcie卡项目中有效，代表每张板卡的id值，从0开始
index：只 在pcie卡项目中有效，代表指定芯片在对应板卡中的编号，从0开始
dev_id：代表当前系统中所有设备的编号值，从0开始
key： 代表系统中每个dev_id 会对应一个唯一的key，用户可以不用关心
ttyACMx：如果device有虚拟串口功能，且device处于正常启动状态（ kernel ），对应的值代表host看见的虚拟串口号。用户可以通过minicom -D /dev/ttyACMx 访问对应device的终端；否则为x，代表当前状态不支持
slot_name：只在pcie卡项目中有效，代表对应pcie卡的槽位名
speed：代表对应设备的usb速度，比如12对应usb1.1，480对应usb2.0，5000对应usb3.0
busnum：代表对应设备在host端分配的usb总线编号
devnum：代表对应设备在host端分配的usb设备地址
mode：代表当前设备处于的模式，以下为mode的项目描述

mode	name	ep_nums	in_nums	out_nums	备注
1	zero + raw	12	6	6	
2	zero + raw + serial	9	4	5	
3	zero + raw + uvc	10	4	6	
4	zero + raw + uvc + serial	7	2	5	
5	zero + raw + hid	10	5	5	
6	zero + raw + hid + serial	7	3	4	

上表中mode代表对应的usb模式值，ep_nums代表rpc+p2p可用端点数, in_nums代表rpc+p2p的d2h方向可用端点数, out_nums 代表h2d方向的可用端点数
status：代表设备当前的状态值，目前状态有：x(未知状态)，bootrom（刷机状态），uboot（uboot状态），kernel（正常启动状态）

2.4 配置信息

2.4.1 生成usbprop.ini文件

2.4.1.1 命令参数

```
# ./desdk set -t usbprop_ini -f <ini_path> -n <dev_nums> [-F {force}] : create
usbprop.ini
```

-f：需要指定生成usbprop.ini文件的路径

-n：为当前系统的device个数总和
-F：为强制模式，该参数用于host没有复位设备功能的场景，比如没有gpio或者smbus

2.4.1.2 使用说明

```
# ./desdk set -t usbprop_ini -f /root/usbprop.ini -n 4
switch all device into usb mode
start all device ...
success to create /tmp/usbprop.ini
```

2.4.1.3 注意事项

1）该命令需要在固件烧录成功后运行，会重启系统，请耐心等待命令结束。如果host没有复位设备功能，需要手动将设备切换到work模式后重启

2）确保-n输入的值等于系统device个数总和，否则会出现错误或者异常

3）如果系统有gpio或者smbus，请尽量不要使用-F，因为-F可能在特殊场景下导致usbprop.ini生成不准确

2.4.2 配置usb模式

2.4.2.1 命令参数

```
# ./desdk set -t usbmode_ini [-i <dev_id>] -m <usb_mode> : config usb mode of
specified device
```

-i： 设备id，如果没有-i，则默认同时配置所有设备

-m：为当usb模式的值，具体值详见3.4表格中的描述

2.4.2.2 使用说明

```
# ./desdk set -t usbmode_ini -i 0 -m 2
Success to set usb mode!!!
```

注：配置完成后可以通过./desdk info -t usb -l -m 查看

2.4.2.3 注意事项

1）不同模式会消耗不同的usb端点数量，导致usb剩余端点数不同，详细看3.4的表格描述

2.4.3 dev_id和gpio的映射关系

在实现libctrl.so库的int PcieReset(int device_id)接口时，由于device_id是一个抽象概念而gpio是真实值，所以需要知道device_id和gpio的映射关系。

2.4.3.1 实现PcieReset接口

```
int PcieReset(int device_id)
{
    switch(device_id)
    {
        case 0:
            gpio_rest(gpio_a);
            break;
        case 1:
            gpio_rest(gpio_d);
            break;
        case 2:
            gpio_rest(gpio_c);
            break;
        case 3:
            gpio_rest(gpio_d);
            break;
        default :
            return -1;
    }
    return 0;
}
```

注意：

1）以上为伪代码，且假设系统有4个device，对应的4个复位gpio分别为gpio_a，gpio_b，gpio_c，gpio_d

2）第一次实现PcieReset接口可以任意填写device_id和gpio的映射关系，以下为伪代码的映射关系

device_id	gpio
0	gpio_a
1	gpio_b
2	gpio_c
3	gpio_d

2.4.3.2 获取正确的映射关系

在4.3.1完成代码编写后生成libctrl.so 库，拷贝到系统中，可以通过ldd desdk来确认libctrl.so路径是否正确

```
# ./desdk info -t usb_id_gpio_map
```

```
switch device 0 int usb mode
Found i2c smbus: /dev/i2c-5
Found i2c smbus: /dev/i2c-5
switch device 0 int work mode
switch device 1 int usb mode
switch device 1 int work mode
switch device 2 int usb mode
switch device 2 int work mode
switch device 3 int usb mode
switch device 3 int work mode
---------------------------
|current_id   |actual_id   |
---------------------------
|0            |1           |
---------------------------
|1            |3           |
---------------------------
|2            |2           |
---------------------------
|3            |0           |
---------------------------
```

通过打印可以看出，正式的映射关系应该是：

device_id	gpio
1	gpio_a
3	gpio_b
2	gpio_c
0	gpio_d

故libctrl.so库的int PcieReset(int device_id)接口应该修改为：

```
int PcieReset(int device_id)
{
    switch(device_id)
    {
        case 1:
            gpio_rest(gpio_a);
            break;
        case 3:
            gpio_rest(gpio_d);
            break;
```

```
        case 2:
            gpio_rest(gpio_c);
            break;
        case 0:
            gpio_rest(gpio_d);
            break;
        default :
            return -1;
    }
    return 0;
}
```

2.4.3.3 更新libctrl.so库

在执行完4.3.2后，按新的映射关系更新PcieReset接口，然后编译新的libctrl.so库，拷贝到系统中即可。

2.5 固件刷机

2.5.1 命令参数

```
# ./desdk download
download usage: i=[0, max_dev_nums)
-r : Reset all device
-i <dev_id1 dev_id2 ...> -r : Reset the specified device
-m usb/work : Switch all device to usb or work mode
-i <dev_id> -m usb/work : Switch specified device to usb or work mode
-b <board_name> -f <ini_path> : [normal mode] Download all device's images in
usbprop.ini
-n <max_dev_nums> -b <board_name> -f <ini_path> : [Factory mode] Download all
device's images without usbprop.ini
-i <dev_id1 dev_id2 ...> -b <board_name> -f <ini_path> : Download the specified
device
-i <dev_id1 dev_id2 ...> -b <board_name> -f <ini_path> -s 0 : Boot the specified
device
```

-r ：代表复位device功能

-i ： 代表指定device的dev_id值，可以输入多个，最大值为系统device个数总和减1

-m：代表 device的系统模式切换功能，usb为刷机模式，work为正常启动模式

-n：代表系统最大的device个数总和，该参数会忽略usbprop.ini，主要用于第一次刷机，生产刷机，或者当前没有usbprop.ini的场景

-b ：代表指定刷机的的板子名称

-f：代表指定刷机包的配置文件路径

2.5.2 使用说明

1) 复位全部device

```
# ./desdk download -r
```

2）复位多个指定device

```
# ./desdk download -i 0 3  -r
```

3）手动切换到所有device的启动模式

```
# ./desdk download -m usb
# ./desdk download -m work
```

4）手动切换到多个指定device的启动模式

```
# ./desdk download -i 0 3 -m usb
# ./desdk download -i 0 3 -m work
```

5）对当前所有设备刷机

```
# ./desdk download -b desdk -f file/config.ini
```

6）对当前多个指定设备刷机

```
# ./desdk download -i 0 3 -b desdk -f file/config.ini
```

7）强制所有设备刷机

```
# ./desdk download -n 4 -b desdk -f file/config.ini
```

该模式主要用于系统没有usbprop.ini的场景强制刷机，比如第一次刷机，生产刷机等，请确保-n参数要等于当前device个数总和，否则可能引起异常

2.5.3 注意事项

1）除了5.2中的第7点外，其他命令都要依赖USB_CONFIG_INI环境变量

2）-i 参数的最大值为系统device总数减1

2.6 文件传输

2.6.1 命令参数

```
# ./desdk transfer
transfer usage: i=[0, max_dev_nums)
-i <dev_id> -u <src_file dst_file>:
    Upload/Send src_file into dst_file of specified dev_id.
-i <dev_id1 dev_id2 ...> -u <src1_file dst1_file src2_file dst2_file ... ...>:
    Upload/Send diff file into diff specified dev_id in same time.
-u <src_file dst_file>:
    Upload/Send src_file into dst_file of all devices in same time.
-i <dev_id> -d <src_file dst_file>:
    Download/Recv dst_file from src_file of specified dev_id.
-i <dev_id> -c <command>:
    Send command into specified dev_id.
-i <dev_id1 dev_id2 ...> -c <command1 command2 ...>:
    Send diff command into diff specified dev_id in same time.
```

```
-c <command>:
    Send command into all devices in same time.

example: max_dev_nums=4
-i 0 -u abc /tmp/abc
-i 0 3 -u abc0 /tmp/abc0 abc3 /tmp/abc3
-u abc /tmp/abc
-i 0 -d /tmp/abc abc
-i 0 -c "cd /tmp; ls"
-i 0 3 -c "cd /tmp; ls" "cd /; ls"
-c "cd /tmp; ls"
```

-i ： 代表指定设备的dev_id值，可以输入多个，最大值为系统device个数总和减1

-u ：代表host发送文件到device

-d：代表host接收文件从device

-c：代表host发送命令到device，并且执行该命令

2.6.2 使用说明

1）host发送文件到指定device

```
# ./desdk transfer -i 0 -u abc /tmp/abc
```

2）host发送不同文件文件到多个指定device

```
./desdk transfer -i 0 3 -u abc0 /tmp/abc0 abc3 /tmp/abc3
```

其中上面代表把host的abc0传到device0的/tmp/abc0中，把abc3传到device3的/tmp/abc3中

3）host发送相同文件到所有device

```
# ./desdk transfer -u abc /tmp/abc
```

4）host接收文件从单个指定device

```
# ./desdk transfer -i 0 -d /tmp/abc abc
```

5）host发送命令到指定device

```
# ./desdk transfer -i 0 -c "cd /tmp; ls"
```

6）host发送命令到多个指定device

```
# ./desdk transfer -i 0 3 -c "cd /tmp; ls" "cd /; ls"
```

其中"cd /tmp"对应dev_id=0的设备，"cd /；ls"对应dev_id=3的设备

7）host发送命令到所有device

```
# ./desdk transfer -c "cd /tmp; ls"
```

2.6.3 注意事项

1）目前只host从device接收文件只支持指定单个设备传输，不支持指定多个设备

2）在host发送文件到多个device的时候，确保-u后面的参数个数匹配

3）该命令依赖USB_CONFIG_INI环境变量

2.7 测试

包含以下功能测试

```
# ./desdk test
test usage:
-t ddr                  : ddr stress test, add -h to get more info
-t usb                  : usb stress test, add -h to get more info
-t flash                : flash stress test, add -h to get more info
```

后续添加生产（product）、老化（aging）等测试

2.7.1 DDR测试

2.7.1.1 命令参数

```
# ./desdk test -t ddr
ddr test usage:
[-i <dev_id>] -p <n> <size> <time> : ddr stress test
[-i <dev_id>] -p 2 pre : pre-proccess for case 2 (replace files)
[-i <dev_id>] -p 2 post : post-proccess for case 2 (restore files)
    dev_id  - one or more device id
    n    - case number: 1 (high bandwidth), 2 (high voltage ripple)
    size - size of ddr test in MB (16 as default)
    time - time of ddr test in second (use 0 for non-stop)
example:
-p 1 16 60 : ddr stress test on all devices (case 1, 16MB, 60s)
-i 0 -p 1 16 60 : ddr stress test on device 0 (case 1, 16MB, 60s)
-p 2 pre : pre-proccess for case 2 on all devices
-p 2 16 3600 : ddr stress test on all devices (case 2, 16MB, 3600s)
-p 2 post : post-proccess for case 2 on all devices
```

-i ：代表指定device的dev_id值，可以输入多个，最小值0，最大值为系统device个数总和减1

-p：启动DDR测试的参数，后面需要输入3个参数，参数分别是：

- n：用例编号（1或2），case 1是高带宽用例，case 2是电流波动大用例
- size：每次拷贝数据大小，单位是MB，默认用16
- time：测试时间，单位是秒

关于case 2要注意：

1. case 2在开始前需要预处理操作（更改DP1000侧的desdk.cfg）并手动复位DP1000（desdk download [-i N] -r）
2. 开始跑之后直到重启DP1000，DSP会一直运行纹波测试（有LED闪灯），此时不要运行其他测试

3. 测试完毕需要后处理操作恢复正常版本（desdk.cfg），并手动复位（desdk download [-i N] -r）

2.7.1.2 使用说明

1）所有设备全部测试case 1（10秒，1分钟，1小时）

```
# ./desdk test -t ddr -p 1 16 10
# ./desdk test -t ddr -p 1 16 60
# ./desdk test -t ddr -p 1 16 3600
```

2）指定设备测试case 1（设备0，设备0和2）

```
# ./desdk test -t ddr -i 0 -p 1 16 60
# ./desdk test -t ddr -i 0 2 -p 1 16 60
```

3）所有设备全部测试case 2（2分钟，1小时）

```
# ./desdk test -t ddr -p 2 pre
# ./desdk download -r

# ./desdk test -t ddr -p 2 16 120
# ./desdk test -t ddr -p 2 16 3600

# ./desdk test -t ddr -p 2 post
# ./desdk download -r
```

2.7.1.3 查看测试结果

1）到设定的时间结束后，如果测试成功会打印

```
DDR Test succeed
```

如果失败会打印

```
DDR Test fail: M / N devices
```

其中M是失败的设备数，N是测试的总设备数

2）在测试的过程中如果想确认是否出错，

case 1关注如下打印中的`cnt_fail=`，表示累计出现过几次比对错误，0表示正常，大于0表示失败（具体情况查找`fail to compare`）

```
Dev0-263146354: 2, success to compare (cnt_fail=0)
```

case 2除了关注上面打印，还要关注如下打印中的`run_cnt=`，一直在增长表示正常，停止增长表示失败

```
Dev0-76173539: 1970-01-01-08:01:21 run_cnt=4005, temp0=51, temp1=46
Dev0-136174889: 1970-01-01-08:02:21 run_cnt=7944, temp0=52, temp1=48
```

2.7.2 flash测试

2.7.2.1 命令参数

```
# ./desdk test -t flash
flash test usage:
[-i <dev_id>] [-p <d2f>/<f2d>/<f2f> <path> <size> <time>] [-c <path>] : flash
stress test
    dev_id  - device id
    d2f  - ddr to flash
    f2d  - direction from flash to ddr
    f2f  - direction from  flash to flash
    path  - path of flash test
    size - size of flash test, unit MB
    time - time of flash test, unit second
example:
-p f2f /root/data 20 200
-i 0 -p f2f /root/data 20 200
-i 0 2 -c /root/data : clean up /root/data of device0 and device2
-c /root/data : clean up /root/data of all device
-i 0 2 -c /root/data : clean up /root/data of device0 and device2
```

-i ：代表指定device的dev_id值，可以输入多个，最大值为系统device个数总和减1

-p：启动flash测试的参数，后面需要输入4个参数，参数分别是：

d2f ：数据从ddr拷贝到flash

f2d：数据从flash拷贝到ddr

f2f：数据从flash拷贝到flash

path：flash数据拷贝的路径

size：每次拷贝数据大小，单位是MB

time：flash测试的时间，单位是秒

-c：代表清除flash测试过程中产生的数据

path： flash数据拷贝的路径，确保和-p中的一致

2.7.2.2 使用说明

1) 所有设备全部测试

```
# ./desdk test -t flash -p f2f /root/data 20 200
```

2) 指定设备测试

```
# ./desdk test -t flash -i 0 2 -p f2f /root/data 20 200
```

3) 清除所有设备测试中产生的临时数据

```
# ./desdk test -t flash -c /root/data
```

4) 清除指定设备测试中产生的临时数据

```
# ./desdk test -t flash -i 0 2 -c /root/data
```

2.7.2.3 测试结果

1) 测试成功

```
success to flash test dev_id=0
success to flash test dev_id=1
```

2) 测试失败

```
fail to flash test dev_id=0
success to flash test dev_id=1
```

2.7.2.4 注意事项

1) 该命令依赖USB_CONFIG_INI环境变量

2.7.3 usb测试

2.7.3.1 命令参数

```
# ./desdk test -t usb
usb test usage:
-f <path> [-p <d2f>/<f2d>/<f2f> <path> <size> <time>]  : usb stress test [parallel
flash stress test]
    path - path of config_p2p.ini
    d2f  - ddr to flash
    f2d  - direction from flash to ddr
    f2f  - direction from  flash to flash
    path  - path of flash test
    size - size of flash test, unit MB
    time - time of flash test, unit second
example:
-f p2p_config.ini
-f p2p_config.ini -p d2f /root/data 10 100
```

-f：代表指定config_p2p.ini配置文件的路径

-p：代表同时启动flash测试的参数，后面需要输入4个参数，参数分别是：

d2f ：数据从ddr拷贝到flash

f2d：数据从flash拷贝到ddr

f2f：数据从flash拷贝到flash

path：flash数据拷贝的路径

size：每次拷贝数据大小，单位是MB

time：flash测试的时间，单位是秒

2.7.3.2 config_p2p.ini 文件使用

```
[device0]
ENABLE=1
TRANSFER_TIME=5
H2D_NUMS= 2
H2D_QUEUE_SIZE= 16 16 16 16 16 16
H2D_MAX_BLOCK_SIZE= 0x20000 0x20000 0x20000 0x20000 0x20000 0x20000
```

```
H2D_LEN= 0x10000 0x11000 0x12000 0x13000 0x14000 0x15000
H2D_CRC= 0 0 0 0 0 0
H2D_DELAY=0 0 0 0 0 0
D2H_NUMS= 2
D2H_QUEUE_SIZE= 16 16 16 16 16 16
D2H_MAX_BLOCK_SIZE= 0x20000 0x20000 0x20000 0x20000 0x20000
D2H_LEN= 0x10000 0x11000 0x12000 0x13000 0x14000
D2H_CRC= 0 0 0 0 0
D2H_DELAY=0 0 0 0 0

[device1]
...
[device2]
...
```

[device0]：设备的节点名，对应dev_id=0

ENABLE：节点使能开关，为0代表关系，1代表打开

TRANSFER_TIME：该节点测试的时间，单位秒

H2D_NUMS：代表host to device方向打开多少通道

H2D_QUEUE_SIZE：代表host to device方向内部缓存多少队列

H2D_MAX_BLOCK_SIZE：代表host to device方向每个数据块最大的内存大小，单位byte

H2D_LEN：代表host to device方向每个数据块的包大小

H2D_CRC：代表数据发送是否需要做crc校验，为0代表不需要，1代表需要

H2D_DELAY：代表每次发送完成后是否需要延时等待特定时间再发送下一包数据，单位微秒

注：

1）D2H和H2D参数函数类似，只是方向不一样

2）很多参数后面有多个数字，每个数字代表不同通道的配置，如果通道为2，则只有前2个有效，后面的将忽略

2.7.3.3 使用说明

1）单独测试usb

```
# ./desdk test -t usb -f config_p2p.ini
```

2）同时测试usb以及flash

```
# ./desdk test -t usb -f config_p2p.ini -p f2f /root/data 20 200
```

2.7.3.4 测试结果

1）测试成功

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 ~~~~~~usb bandwidth test: ~~~~~~
 dev_id=0 h2d_speed=18.920057MB/s    d2h_speed=-nan  MB/s
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

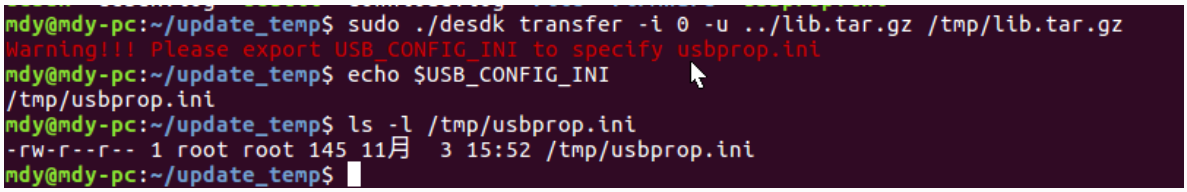
注：usb如果失败可能导致进程本科或者阻塞

2.7.3.5 注意事项

1) 默认config_p2p.ini只使能了device0，如果要测试多个设备，请做对应的修改

2.8 常见问题总结

2.8.1 已经export USB_CONFIG_INI，但是仍然提示找不到



原因：由于账号切换导致的，export的是在本地账号执行的，但是执行工具的时候通过sudo 切换到了root账号，所以导致在root账号环境下无法找到USB_CONFIG_INI

解决方案：

方案一：直接sudo su 切换到root找好执行export 以及 desdk工具

方案二：sudo USB_CONFIG_INI=/tmp/usbprop.ini ./desdk xxx xxx

2.8.2 提示比较设备数失败 fail to compare nums of device

原因：这个错误原因比较多，总的来说就是检测到的USB端口数与命令中不符。

解决方案：需要用lsusb检查匹配的USB设备数，再检查命令中指定的设备数，如-i，-n等参数。