hcRTOS USB gadget Zero 驱动说明

概述

hcRTOS SDK 提供自定义usb gadget驱动，其中使用的框架类似Linux 的USB Gadget driver, 也一样提供 Zero 驱动来作为demo展示

> Zero驱动包含两个 BULK 端点, 分别为一个 IN 端点，一个 OUT端点。 基于这两个（由底层提供的）端点，g_zero 驱动实现了两个 configuration。
>
> - 第一个 configuration 提供了 **sink/source**功能：两个端点一个负责输入，一个负责输出，其中输出的内容根据设置可以是全0，也可以是按照某种算法生成的数据。
> - 另一个 configuration 提供了 **loopback** 接口， IN 端点负责把从 OUT 端点收到的数据反馈给 Host.

如源码所示(路径: `components\kernel\source\drivers\usb\gadget`) , hcRTOS的usb驱动框架参照 Linux kernel，hcRTOS的usb gadget驱动也是通过 usb composite层和function层进行配置, 进而完成usb gadget 驱动的实现.

关于usb gadget驱动框架以及API说明, 请见Linux 官方说明: https://www.kernel.org/doc/html/v4.17/driver-api/usb/gadget.html

hcRTOS 环境

menuconfig配置

使用 `make menuconfig` , 打开以下编译开关来使能

- `BR2_PACKAGE_PREBUILTS_USBGADGETDRIVER`
- `CONFIG_USB_GADGET_VENDOR`
- `CONFIG_USB_ZERO`
- `CONFIG_CMDS_USB_GADGET_ZERO` // 备注: 这只是测试命令, 并非必需的

完成上述配置后, 执行以下命令进行编译

```
make kernel-rebuild cmds-rebuild all
```

板端执行命令

上电后, 执行以下命令, 就能让对应的usb端口配置为 zero 驱动.

```
usb g_zero -p 0 ## 使用usb#0
usb g_zero -p 1 ## 使用usb#1
```

此时使用USB线材连接到PC上，如果是linux PC平台的话, 可以使用 `lsusb` 可以查看到

```
> lsusb
Bus 001 Device 002: ID 0525:a4a0 Netchip Technology, Inc. Linux-USB "Gadget Zero"
```

PC端测试程序

程序程序源码

为了测试这个Zero驱动, 以下用libusb 编写一个简单的测试程序 (**g_zero_test.c** 和 **Makefile**)

```
#include <stdio.h>
#include <string.h>
#include <libusb.h>

#define TIMEOUT 2000
#define STR_LEN 20

int main(int argc, char *argv[]) {
    int ret = 0;
    libusb_device_handle *handle;
    libusb_device **list;
    libusb_device *usbdev;
    struct libusb_device_descriptor dev_desc;
    struct libusb_config_descriptor *config_desc;
    const struct libusb_endpoint_descriptor *ep_desc;
```

```c
    unsigned char ep_bulkin = 0 , ep_bulkout = 0;
    int i = 0, is_match = 0;
    int ep_cnt;
    char send_data[STR_LEN] = {0,};
    char recv_data[STR_LEN] = {0,};
    int transfered = 0;
    int count = 0;

    ret = libusb_init(NULL);
    if (ret < 0) {
        fprintf(stderr, "libusb_init failed, ret(%d)\n", ret);

        return -1;
    }

  //  libusb_set_debug(NULL, LIBUSB_LOG_LEVEL_DEBUG);

    // get usb device list
    ret = libusb_get_device_list(NULL, &list);
    if (ret < 0) {
        fprintf(stderr, "libusb_get_device_list failed,"
                "ret(%d)\n", ret);

        goto get_failed;
    }

    /* print/check the matched device */
    while ((usbdev = list[i++]) != NULL) {
        libusb_get_device_descriptor(usbdev, &dev_desc);

#if 1
        printf("usb-%d: pid(0x%x), vid(0x%x)\n",
               i++,
               dev_desc.idVendor,
               dev_desc.idProduct);
#endif

        if (dev_desc.idVendor == 0x0525 &&
                dev_desc.idProduct == 0xa4a0) {
            printf("match, break!\n");
            is_match = 1;
            break;
        }
    }

    if (!is_match) {
        fprintf(stderr, "no matched usb device ... \n");

        goto match_fail;
```

```
65      }
66
67      /* open usb device */
68      ret = libusb_open(usbdev, &handle);
69      if (ret < 0) {
70          fprintf(stderr, "libusb_open failed. ret(%d)\n", ret);
71
72          goto open_failed;
73      }
74
75      printf("this usb device has %d configs, "
76              "but still use default config 0\n",
77              dev_desc.bNumConfigurations);
78
79      /* get config descriptor */
80      ret = libusb_get_config_descriptor(usbdev, 0, &config_desc);
81      if (ret < 0) {
82          fprintf(stderr, "get config descriptor failed ... \n");
83          goto configdesc_fail;
84      }
85
86      printf("the config has %d interface ... "
87              "just use the 1st interface this time\n",
88              config_desc->bNumInterfaces);
89
90      ep_cnt = config_desc->interface->altsetting[0].bNumEndpoints;
91
92      printf("this interface has %d endpoint\n", ep_cnt);
93
94      /* get bulk in/out ep */
95      for (i=0; i<ep_cnt; i++) {
96          ep_desc = &config_desc->interface->altsetting[0].endpoint[i];
97          if ((ep_desc->bmAttributes & LIBUSB_TRANSFER_TYPE_MASK) &
    LIBUSB_TRANSFER_TYPE_BULK) {
98              if ((ep_desc->bEndpointAddress & LIBUSB_ENDPOINT_DIR_MASK) &
    LIBUSB_ENDPOINT_IN)  {
99                  if (!ep_bulkin) {
100                     ep_bulkin = ep_desc->bEndpointAddress;
101                 }
102             } else {
103                 if (!ep_bulkout) {
104                     ep_bulkout = ep_desc->bEndpointAddress;
105                 }
106             }
107
108             printf("ep_addr = 0x%x\n", ep_desc->bEndpointAddress);
109         }
110     }
111
```

```
112     if (ep_bulkin) {
113         printf("yes, got 1 bulk in endppint!\n");
114     }
115 
116     if (ep_bulkout) {
117         printf("yes, got 1 bulk out endppint!\n");
118     }
119 
120  //   printf("reset usb device\n");
121  //   libusb_reset_device(handle);
122 
123     printf("check usb interface0 : %d\n",
    libusb_kernel_driver_active(handle, 0));
124 
125     /* claim the interface */
126     printf("claim usb interface\n");
127     ret = libusb_claim_interface(handle, 0);
128     if (ret < 0) {
129         printf("claim usb interface failed, detach and try again\n");
130         libusb_detach_kernel_driver(handle, 0);
131 
132         ret = libusb_claim_interface(handle, 0);
133         if(ret < 0){
134             printf("claim usb interface failed ... ret(%d)\n", ret);
135             goto claim_failed;
136         }
137     }
138 
139     printf("start bulk transfer ... \n");
140     /* use bulk trandfer data directly */
141     while (count++ < 20) {
142         snprintf(send_data, STR_LEN, "hello,world - %d", count);
143         ret = libusb_bulk_transfer(handle, ep_bulkout, (unsigned
    char*)send_data, sizeof(send_data),
144                 &transfered, TIMEOUT);
145 
146         if (ret == 0) {
147             ret = libusb_bulk_transfer(handle, ep_bulkin, (unsigned char
    *)recv_data,
148                     sizeof(recv_data), &transfered, TIMEOUT);
149 
150             if (ret == 0) {
151                 printf("recv: %s\n", recv_data);
152             } else {
153                 printf("bulk in failed, ret(%d), transfered(%d)\n",
154                         ret, transfered);
155             }
156         } else {
157             printf("bulk out failed, ret(%d), transfered(%d)\n",
```

```
                    ret, transfered);
        }
    }

    // libusb_open(dev, &handle);
    libusb_close(handle);

    libusb_free_device_list(list, 1);

    libusb_exit(NULL);

    return 0;

configdesc_fail:
claim_failed:
    libusb_close(handle);

open_failed:
match_fail:
    libusb_free_device_list(list, 1);

get_failed:
    libusb_exit(NULL);

    return ret;
}
```

Makefile

```
CC=gcc
CFLAGS=-g -Wall
CFLAGS+=`pkg-config --cflags libusb-1.0`
LDFLAGS=`pkg-config --libs libusb-1.0`

target=usbtest
objs=$(patsubst %.c, %.o, $(wildcard *.c))

all:$(target)

$(target):$(objs)
    $(CC) $^ -o $@ $(LDFLAGS)

.c.o:
    $(CC) -c $< $(CFLAGS)
```

```
16
17  .PHONY:
18      clean
19
20  clean:
21      rm *.o $(target) -rf
```

H3 执行示例

建议执行命令之前, 使用 sudo, 要不会提示权限不足的错误提示

```
1  $ sudo ./usbtest
2  usb-1: pid(0x525), vid(0xa4a0)
3  match, break!
4  this usb device has 2 configs, but still use default config 0
5  the config has 1 interface ... just use the 1st interface this time
6  this interface has 2 endpoint
7  ep_addr = 0x81
8  ep_addr = 0x1
9  yes, got 1 bulk in endppint!
10 yes, got 1 bulk out endppint!
11 check usb interface0 : 0
12 claim usb interface
13 start bulk transfer ...
14 recv: hello,world - 1
15 recv: hello,world - 2
16 recv: hello,world - 3
17 recv: hello,world - 4
18 recv: hello,world - 5
19 recv: hello,world - 6
20 recv: hello,world - 7
21 recv: hello,world - 8
22 recv: hello,world - 9
23 recv: hello,world - 10
24 recv: hello,world - 11
25 recv: hello,world - 12
26 recv: hello,world - 13
27 recv: hello,world - 14
28 recv: hello,world - 15
29 recv: hello,world - 16
30 recv: hello,world - 17
31 recv: hello,world - 18
32 recv: hello,world - 19
```

```
33  recv: hello,world - 20
```