Linux USB
开发指南

版本号: 2.5
发布日期: 2023.03.06

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.6.30	AWA1458	初始版本。
1.1	2020.10.27	AWA1458	1. 添加 Gadget 配置；2. 添加端点修改；3. 更换清晰图片等。
1.2	2021.01.18	AWA1458	添加 USB Standby 注意事项。
1.3	2021.03.10	XAA0190	1. 更换清晰图片；2. 添加使用范围；3. 添加功能配置说明等。
1.4	2022.05.09	AWA1736	修改排版格式。
1.5	2022.05.19	AWA1927	1. 添加支持 Type C 功能配置说明；2. 修复小机作为 ADB+Mass Storage 时 PC 无法格式化问题。
1.6	2022.06.22	AWA1736	1. 删除行末空格；2. 修复修改记录格式；3. 修复标点符号为中文格式；4. 修复代码段对齐方式。
1.7	2022.07.05	AWA1927	增加 FAQ 关于停止 ADB 配置其他 gadget 功能时异常报错解决办法。
1.8	2022.11.04	AWA1881	1. 新增 Linux-5.10、Linux-5.15 内核相关配置说明；2. 使用 Linux-5.4 的设备树配置替换旧版的 linux-4.9 配置。
2.0	2022.11.07	AWA1736	1. 迭代版本以 Linux-5.x 为文档基线；2. 增加 “模块配置释义” 小节，描述配置含义及使用；3. 优化文档层级结构；4. 优化设备树及 board.dts 的注意事项描述；5. 优化 menuconfig 说明，增加 bsp 仓库配置；6. 优化端点配置的描述。
2.1	2022.11.21	AWA1736	1. 修改路径格式；2. 增加 “模块硬件介绍” 小节；3. 增加 “驱动能力调节” 小节；4. 增加 “Gadget 功能切换” 小节；4. 增加 “USB 常见问题思路分析” 小节。
2.2	2022.11.23	AWA1736	1. 调整 “调试节点” 小节；2. 增加 “日志分析” 小节。
2.3	2022.11.24	AWA1927	1. 增加 “OTG 配置” 小节；2. 增加 “manager 逻辑分析” 小节；3. 增加 “usb 模块 ko 化配置与使用” 小节。
2.4	2022.12.05	AWA1881	新增常用调试节点。

2.5	2023.03.06	AWA1881	1. 新增 “ACIN 配置方法” 说明；2. 新增 “BOOST 供电配置方法” 说明；3. 新增 “插入 OTG U 盘（3.0）无法识别” 修复方法说明；4. 新增 “USB Standby 调试方法” 说明。

目　录

插　图

1 前言

1.1 文档简介

介绍 USB 模块配置和调试方法。

1.2 目标读者

USB 模块开发、维护人员。

1.3 适用范围

表 1-1: 适用产品列表

内核版本	驱动文件
Linux-4.9	drivers/usb/*
Linux-5.4	drivers/usb/*
Linux-5.10	drivers/usb/*、bsp/drivers/usb
Linux-5.15	drivers/usb/*、bsp/drivers/usb

2 模块介绍

2.1 模块功能介绍

USB 有主机功能和从设备功能。做主机时，能连接 U 盘、USB 鼠标等 USB 设备；做从设备时，具有 ADB 调试等从设备功能。

AW SOC 通常内置多个 USB 控制器，不同控制器互相独立，每套 USB2.0 控制器都包含 EHCI 和 OHCI，能够向下兼容多种协议。关于每个平台的具体的 USB 套数、参数、特性、模块结构图等，请参考发布文档中 **《XXX_User_Manual_Vx.x.pdf》**对应模块的描述。

2.2 相关术语介绍

表 2-1: 术语介绍

术语	说明
USB	Universal Serial Bus, 通用串行总线
OTG	On-The-Go
ADB	Android Debug Bridge，Android 调试桥
Gadget	小配件
HCD	Host Controller Driver，主机控制器驱动
UDC	USB Device Controller, USB 设备控制器
HCI	Host Controller Interface，主机控制器接口
EHCI	Enhanced Host Controller Interface，增强型主机控制器接口
OHCI	Open Host Controller Interface，开放式主机控制器接口

2.3 模块硬件介绍

USB 模块使用 USB0、USB1、USB2、USB3 的形式来标记 USB 接口。

其中 USB0 默认为 OTG 口，默认使用 Micro-B/type-C 接口，其他的默认为 HOST 接口，默认使用 Type-A 接口。烧录程序只能使用 0 口，在产品功能定义上需要注意区别。

USB 模块的配置涉及到了信号引用、pin 脚配置、供电公职等，需要结合硬件原理图，具体请参考

发布文档中 **《XXX 硬件设计指南.pdf》**对应 USB 模块的描述。

2.4 模块配置介绍

2.4.1 模块配置释义

AW USB 相关的设备树配置列表及释义如下表所示。

Table: 配置释义

配置	说明
device_type	控制器名称，如”usbc0”，”usbc01”……
usb_port_type	usb0 默认的模式，仅加载 usb0 时有效 0：devcie only 模式 1：host only 模式 2：otg 模式（device/host 自动检测切换）
usb_detect_type	usb0 OTG role 检测源，仅 usb_port_type=2 时有效 0：none，不做检测 1：vbus/id，读取 vbus 和 id 合并检测，一般用于 micro 口 2：pmu，读取 pmu 寄存器检测，一般用于 type-c 口 * 注：该配置在 linux4.9 中不做检测，默认使用 vbus/id 检测，在 linux-5.4 及以上版本新增 pmu 检测后才做区分
usb_detect_mode	usb0 OTG 状态检测模式，仅 usb_port_type=2 时有效 0：thread scan，线程轮询 1：gpio interrupt，id 中断触发
usb_id_gpio	USB ID pin 脚配置，检测插入设备是 host 还是 device 当 usb_detect_type=pmu 时该引脚配置可删除 使用方法参考《Linux_gpio_ 开发指南》 配置参数参考具体产品的原理图
usb_det_vbus_gpio	USB DET_VBUS pin 脚配置，用于检测 vbus 是否被拉高 当配置为”axp_ctrl” 时，表示 axp 提供 使用方法参考《Linux_gpio_ 开发指南》 配置参数参考具体产品的原理图
det_vbus_supply	属性引用，当 usb_det_vbus_gpio=”axp_ctrl” 时， 由 PMU 模块提供节点用于引用
drvvbus-supply	属性引用，可由 vbus 节点提供，也可由 PMU 模块提供
usb_wakeup_suspend	USB standby 远程唤醒（满足所有条件才能真正生效） 0：super standby 1：usb standby
wakeup-source	设置为唤醒源，需要配合 usb_wakeup_suspend 一起使用
hci_ctrl_no	控制器编号，如 0，1，2……

usb_regulator_io	linux-3.10 后不再使用。
usb_hsic_used	linux-3.10 后不再使用。
usb_hsic_ctrl	linux-3.10 后不再使用。
usb_hsic_rdy_gpio	linux-3.10 后不再使用。
usb_hsic_regulator_io	linux-3.10 后不再使用
usb_luns	linux-3.10 后不再使用。
rndis_wceis	linux-3.10 后不再使用。
usb_serial_unique	linux-4.9 后不再使用。
usb_serial_number	linux-4.9 后不再使用。
usb_host_init_state	linux-4.4 后不再使用。
usb_drv_vbus_gpio	linux-4.9 后不再使用。
usb_drvvbus_en_gpio	linux-5.4 后新增，用于支持 acin 方案。
usb_gma340_oe_gpio	linux-5.15 后新增，用于仅支持 2.0 控制器的标准 Type-C。
vbusin-supply	属性引用，由 PMU 模块提供。

2.4.2 Device Tree 配置说明

2.4.2.1 路径定义

设备树中定义的是该类芯片对应于 IC 规格的所有配置，设备树文件路径如下：

- linux-5.10 之前：

> 32 位平台：kernel/linux-x.x/arch/arm/boot/dts/{CHIP}.dtsi
> 64 位平台：kernel/linux-x.x/arch/arm64/boot/dts/sunxi/{CHIP}.dtsi

- linux-5.10（含 linux-5.10）之后：

> 32/64 位平台：bsp/configs/linux-x.x/{CHIP}.dtsi

说明

1.linux-x.x 为内核版本，如 linux-5.15；
2.{CHIP}.dtsi 为具体芯片型号，如 sun50iw10p1.dtsi。

2.4.2.2 配置示例

设备树配置示例参考如下所示：（以 linux-5.4 为例进行说明）

- USB0 配置（OTG 接口类型）

```
usbc0: usbc0@10 {
    device_type = "usbc0";
    compatible = "allwinner,sunxi-otg-manager";
    usb_port_type = <2>;
    usb_detect_type = <1>;
    usb_detect_mode = <0>;
    usb_id_gpio;
    usb_det_vbus_gpio;
    usb_wakeup_suspend = <0>;
    status = "disabled";
};

udc: udc-controller@5100000 {
    compatible = "allwinner,sunxi-udc";
    reg = <0x0 0x05100000 0x0 0x1000>, /*udc base*/
          <0x0 0x00000000 0x0 0x100>,  /*sram base*/
          <0x0 0x05200000 0x0 0x1000>; /*usb1 base, for common circuit*/
    interrupts = <GIC_SPI 32 IRQ_TYPE_LEVEL_HIGH>; /*设备使用的中断*/
    clocks = <&ccu CLK_BUS_OTG>, <&ccu CLK_USB_PHY0>; /*设备使用的时钟*/
    clock-names = "bus_otg", "phy";
    resets = <&ccu RST_BUS_OTG>, <&ccu RST_USB_PHY0>;
    reset-names = "otg", "phy";
    status = "disabled"; /*是否使能该设备*/
};

ehci0: ehci0-controller@5101000 {
    compatible = "allwinner,sunxi-ehci0";
    reg = <0x0 0x05101000 0x0 0xFFF>, /*ehci0 base*/
          <0x0 0x00000000 0x0 0x100>,  /*sram base*/
          <0x0 0x05100000 0x0 0x1000>, /*otg base*/
          <0x0 0x07010250 0x0 0x10>,   /*prcm base, for usb standby*/
          <0x0 0x03001000 0x0 0x20>,   /*ccmu base, for common circuit*/
          <0x0 0x05200000 0x0 0x1000>; /*usb1 base, for common circuit*/
    interrupts = <GIC_SPI 30 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_BUS_EHCI0>, <&ccu CLK_USB_PHY0>;
    clock-names = "bus_hci", "phy";
    resets = <&ccu RST_BUS_EHCI0>, <&ccu RST_USB_PHY0>;
    reset-names = "hci", "phy";
    hci_ctrl_no = <0>; /*主机控制器的序列*/
    status = "disabled";
};

ohci0: ohci0-controller@5101400 {
    compatible = "allwinner,sunxi-ohci0";
    reg = <0x0 0x05101400 0x0 0xFFF>,  /*ohci0 base*/
          <0x0 0x00000000 0x0 0x100>,  /*sram base*/
          <0x0 0x05100000 0x0 0x1000>, /*otg base*/
          <0x0 0x07010250 0x0 0x10>,   /*prcm base, for usb standby*/
          <0x0 0x03001000 0x0 0x20>,   /*ccmu base, for common circuit*/
          <0x0 0x05200000 0x0 0x1000>; /*usb1 base, for common circuit*/
    interrupts = <GIC_SPI 31 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_BUS_OHCI0>, <&ccu CLK_USB_OHCI0>, <&ccu CLK_USB_PHY0>;
    clock-names = "bus_hci", "ohci", "phy";
    resets = <&ccu RST_BUS_OHCI0>, <&ccu RST_USB_PHY0>;
    reset-names = "hci", "phy";
    hci_ctrl_no = <0>;
    status = "disabled";
```

```
};
```

说明

详细的 otg 配置及逻辑可参考 “OTG 配置” 小节。

- USB1 配置（HOST 接口类型）

```
usbc1: usbc1@11 {
    device_type = "usbc1";
    usb_wakeup_suspend = <0>;
    status = "disabled";
};

ehci1: ehci1-controller@5200000 {
    compatible = "allwinner,sunxi-ehci1";
    reg = <0x0 0x05200000 0x0 0xFFF>, /*ehci1 base*/
          <0x0 0x00000000 0x0 0x100>, /*sram base*/
          <0x0 0x05100000 0x0 0x1000>, /*otg base*/
          <0x0 0x07010250 0x0 0x10>; /*prcm base, for usb standby*/
    interrupts = <GIC_SPI 33 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_BUS_EHCI1>, <&ccu CLK_USB_PHY1>;
    clock-names = "bus_hci", "phy";
    resets = <&ccu RST_BUS_EHCI1>, <&ccu RST_USB_PHY1>;
    reset-names = "hci", "phy";
    hci_ctrl_no = <1>;
    status = "disabled";
};

ohci1: ohci1-controller@5200400 {
    compatible = "allwinner,sunxi-ohci1";
    reg = <0x0 0x05200400 0x0 0xFFF>, /*ohci1 base*/
          <0x0 0x00000000 0x0 0x100>, /*sram base*/
          <0x0 0x05100000 0x0 0x1000>, /*otg base*/
          <0x0 0x07010250 0x0 0x10>; /*prcm base, for usb standby*/
    interrupts = <GIC_SPI 34 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_BUS_OHCI1>, <&ccu CLK_USB_OHCI1>, <&ccu CLK_USB_PHY1>;
    clock-names = "bus_hci", "ohci", "phy";
    resets = <&ccu RST_BUS_OHCI1>, <&ccu RST_USB_PHY1>;
    reset-names = "hci", "phy";
    hci_ctrl_no = <1>;
    status = "disabled";
};
```

2.4.2.3 注意事项

1. 在 linux-4.9 中，节点名称中的地址需要加上 “0x”，如：udc:udc-controller@0x05100000。
2. 在 linux-4.9 中，clocks 节点需要参考时钟树进行配置，以 udc 为例。

```
clocks = <&clk_usbphy0>, <&clk_usbotg>, <&clk_usbehci1>, <&clk_usbphy1>;
```

3. 在 linux-4.9 中，reg 节点 ohci base、ehci base 均填写基地址，简单示例如下。

```
ehci1: ehci1-controller@0x5200000 {
    compatible = "allwinner,sunxi-ehci1";
    reg = <0x0 0x05200000 0x0 0xFFF>, /*此处填hci1 base*/
    ......
};

ohci1: ohci1-controller@0x5200400 {
    compatible = "allwinner,sunxi-ohci1";
    reg = <0x0 0x05200000 0x0 0xFFF>, /*此处填hci1 base*/
    ......
};
```

4. 配置项 reg 最多有六个参数, 按照 index 顺序如下：
   (1)udc/ehciX/ohciX base: 填写控制器 base 及 length，Linux-4.9 参考注意事项 3 填写；
   (2)sram base: 填写 sram base 及 length，一般无需配置；
   (3)otg base: 填写 USB0 的 udc base 及 length，USB0 无需配置；
   (4)prcm base: 仅部分平台（如 sun50iw10）需要配置，其他平台一般无需配置；
   (5)ccmu base: 仅部分平台（如 sun50iw10）需要配置，其他平台一般无需配置；
   (6)usb1 base: 仅部分平台（如 sun50iw10）需要配置，其他平台一般无需配置。

2.4.3 board.dts 配置说明

2.4.3.1 路径定义

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

> device/config/chips/$IC/configs$/{BOARD}/${内核版本}/board.dts

2.4.3.2 配置示例

- Vbus 配置

```
reg_usb1_vbus: usb1-vbus {
    compatible = "regulator-fixed";
    regulator-name = "usb1-vbus";
    regulator-min-microvolt = <5000000>;
    regulator-max-microvolt = <5000000>;
    regulator-enable-ramp-delay = <1000>;
    gpio = <&r_pio PL 8 GPIO_ACTIVE_HIGH>;
    enable-active-high;
};
```

- USB0 配置

```
&usbc0 {
    device_type = "usbc0";
    usb_port_type = <0x2>;
    usb_detect_type = <0x1>;
    usb_detect_mode = <0x0>;
    usb_id_gpio = <&pio PH 8 GPIO_ACTIVE_HIGH>;
    enable-active-high;
    usb_det_vbus_gpio  = "axp_ctrl";
    det_vbus_supply = <&usb_power_supply>;
    usb_wakeup_suspend = <0>;
    status = "okay";
};

&udc {
    det_vbus_supply = <&usb_power_supply>
    status = "okay";
}

&ehci0 {
    drvvbus-supply = <&reg_drivevbus>;
    status = "okay";
};

&ohci0 {
    drvvbus-supply = <&reg_drivevbus>;
    status = "okay";
};
```

- USB1 配置

```
&usbc1 {
    device_type = "usbc1";
    usb_wakeup_suspend = <0>;
    status = "okay";
};

&ehci1 {
    drvvbus-supply = <&reg_usb1_vbus>;
    status = "okay";
};

&ohci1 {
    drvvbus-supply = <&reg_usb1_vbus>;
    status = "okay";
};
```

说明

若使用 usb standby 模式，需同时满足以下条件：

1、IC 支持远程唤醒，具体参考 IC 对应的《XXX_User_Manual_Vx.x.pdf》；

2、硬件支持远程唤醒，硬件部分需严格按照《硬件设计文档》设计；

3、系统支持远程唤醒，具体参考《Linux_Standby_ 开发指南》；

4、添加属性 “wakeup-source”，并配置 usb_wakeup_suspend = 1。

2.4.3.3 注意事项

1. 在 linux-4.9 中，usb_id_gpio 节点书写方式略有差异。

```
usb_id_gpio = <&pio PH 8 0 0 0xffffffff  0xffffffff  >;
```

2. usb_det_vbus_gpio 有两种配置方式：

- 当配置为 axp_ctrl 控制时，需要添加 det_vbus_supply 属性，引用的节点由 PMU 模块提供。

```
usb_det_vbus_gpio = "axp_ctrl"; //添加至usbc0节点
det_vbus_supply  = <&usb_power_supply>; //添加至udc节点
```

- 当配置为 gpio 控制时，只需要直接配置即可。

```
usb_det_vbus_gpio = <&pio PH 8 GPIO_ACTIVE_HIGH>; //添加至usbc0节点
```

3. drvvbus-supply 属性引用有两种来源，可通过引用 Vbus 或 PMU 模块提供的节点完成。

```
PMU引用：
drvvbus-supply  = <&reg_drivevbus>; //添加至ehciX/ohciX节点
Vbus引用：
drvvbus-supply = <&reg_usb1_vbus>;//添加至ehciX/ohciX节点
```

4. 一般在 dtsi 中应该将 status 置为 “disabled”，在板级配置中置为 “okay”。

2.4.4 kernel menuconfig 配置说明

在 SDK 根目录下，执行./build.sh menuconfig，进入配置主界面，并按以下步骤操作：

- linux-5.10 以下版本此步骤跳过，linux-5.10 及以上版本由于独立仓库的原因，需执行以下操作（直接按 “/” 即可进入搜索）。
  (1) 搜索 “CONFIG_USB_SUPPORT”，并打开配置项（USB 功能依赖项）；
  (2) 搜索 “CONFIG_USB”，并打开配置项（Host 驱动依赖项）；
  (3) 搜索 “CONFIG_USB_GADGET”，并打开配置项（Device 驱动依赖项）；
  (4) 按照下面指示开启对应的配置。
  注：该步骤已配的后续无需再重复配置。

```
Allwinner BSP  --->
  Device Drivers  --->
    USB Host Controller Drivers  --->
      <*> Allwinner EHCI HCD
      <*> Allwinner OHCI HCD
      <*> Allwinner USB Host Controller
```

```
    <*>  Allwinner USB HCI
    <*>    Allwinner USB EHCI0
    <*>    Allwinner USB EHCI1
    <*>    Allwinner USB OHCI0
    <*>    Allwinner USB OHCI1
  USB Device Drivers  --->
    <*> Allwinner USB2.0 Dual Role Controller support  --->
      <*>  Allwinner USB2.0 Manager
      <*>  Allwinner USB driver debug message
      <*>  Allwinner USB driver use adb source
    <*> Allwinner USB Peripheral Controller
```

- 选择 SCSI SCSI device support 选项，并进入下一级配置：
  -打开 SCSI device support
  -打开 legacy /proc/scsi/ support
  -打开 SCSI disk support
  -打开 SCSI low-level drivers

- 选择 Device Drivers 选项进入下一级配置，如下图所示。

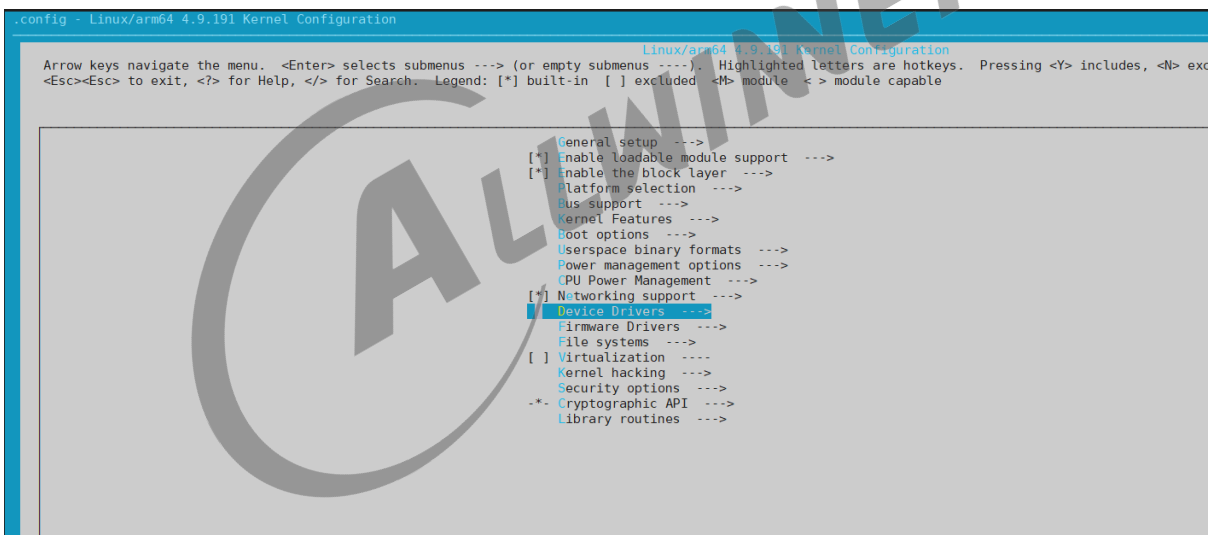

图 2-1: Device Drivers 选项配置

- 选择 USB support 选项，进入下一级配置，如下图所示。

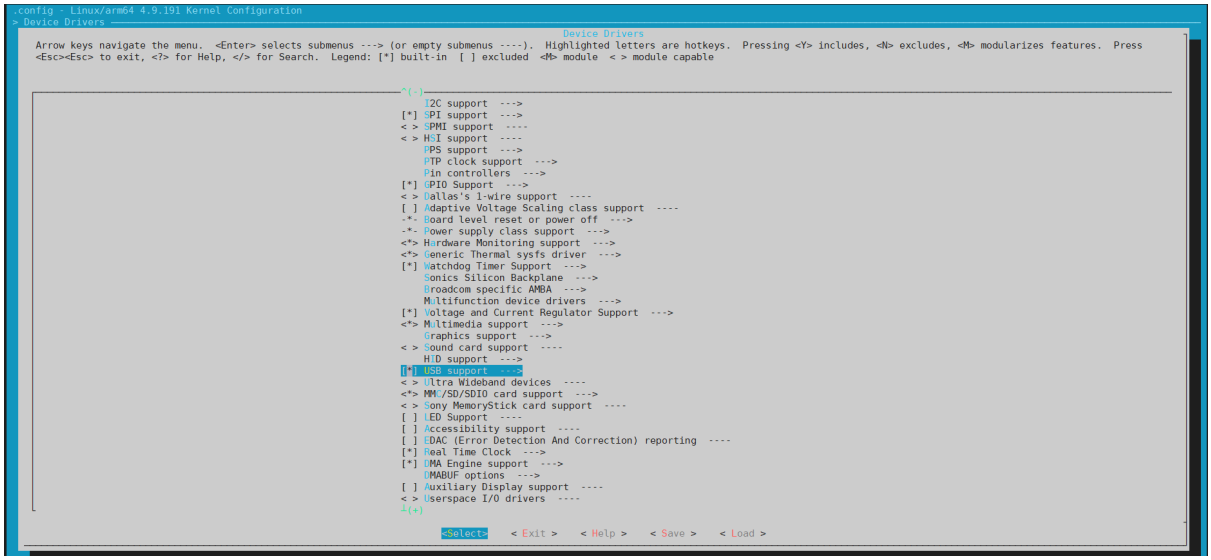

图 2-2: USB Support 选项配置

- 打开如下两图所示的详细配置，如下图所示 (linux-5.10 及以上在第一步已经配置的，无需重复配置)。

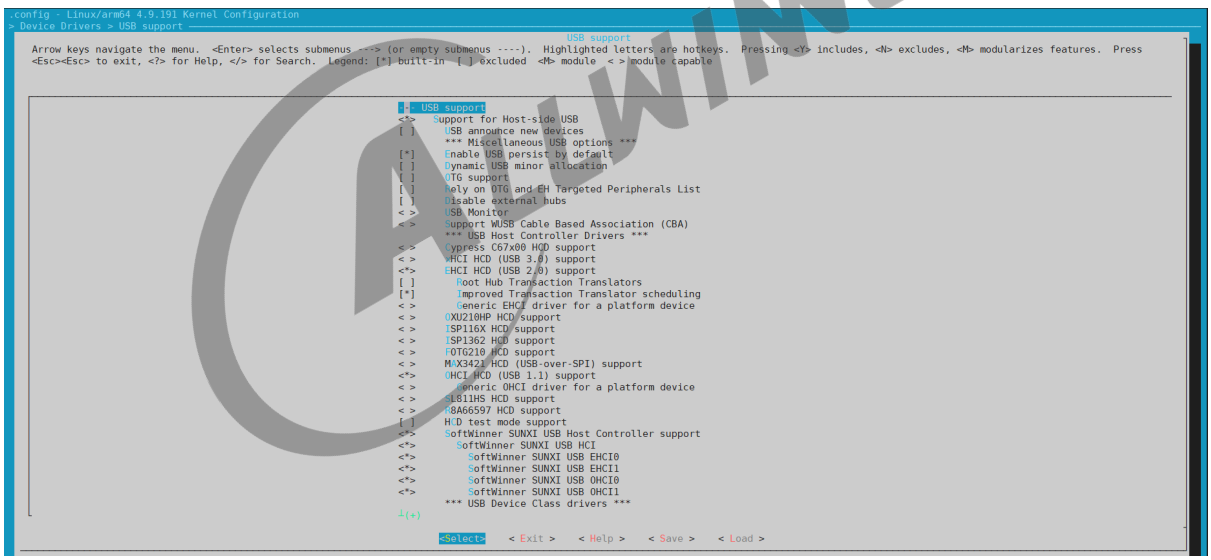

图 2-3: USB Support 详细配置 1

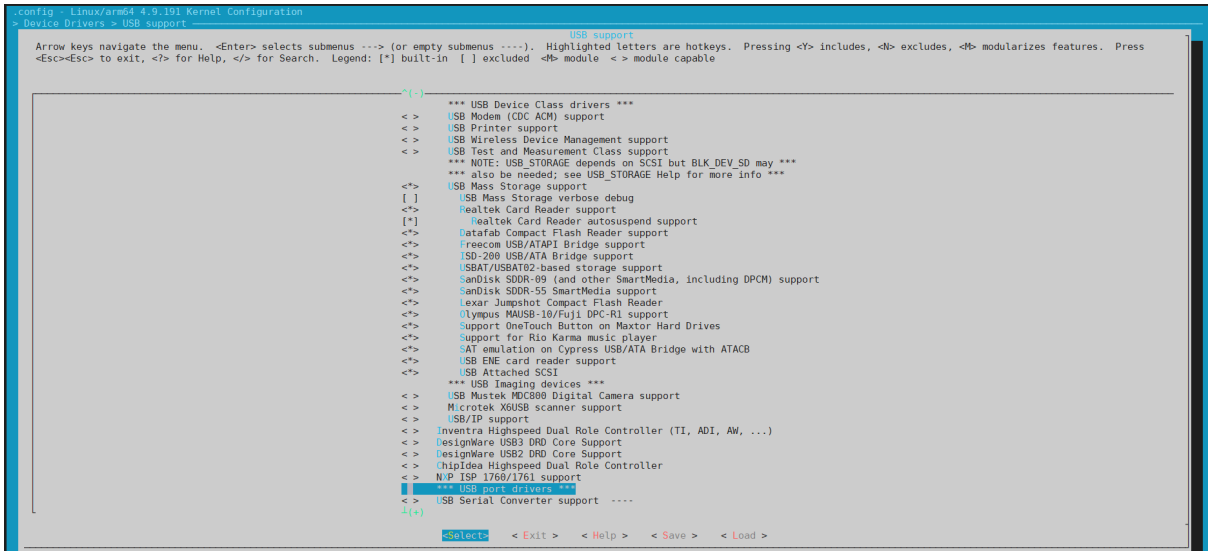

图 2-4: USB Support 详细配置 2

- 选择 USB Gadget Support，进入下一级配置，如下图所示 (linux-5.10 及以上在第一步已经配置的，无需重复配置)。

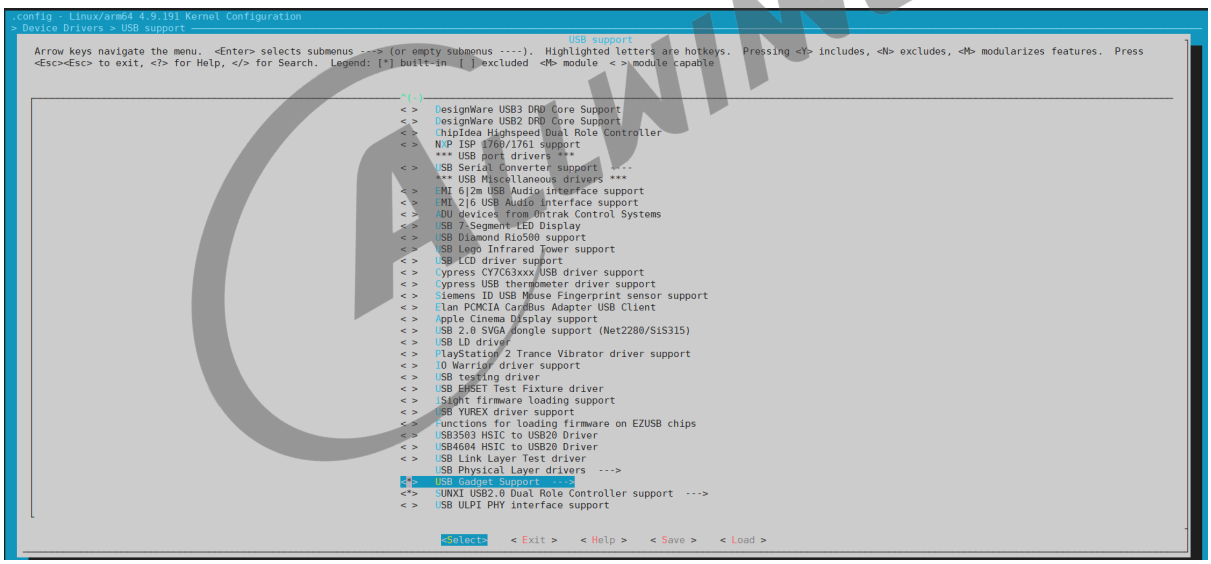

图 2-5: USB Gadget Support 选项配置

- 打开下图的选项，并在对应配置中打开所需的功能性配置, 如: 需要存储功能时, 需打开下图中的“mass storage” 配置, 如下图所示。

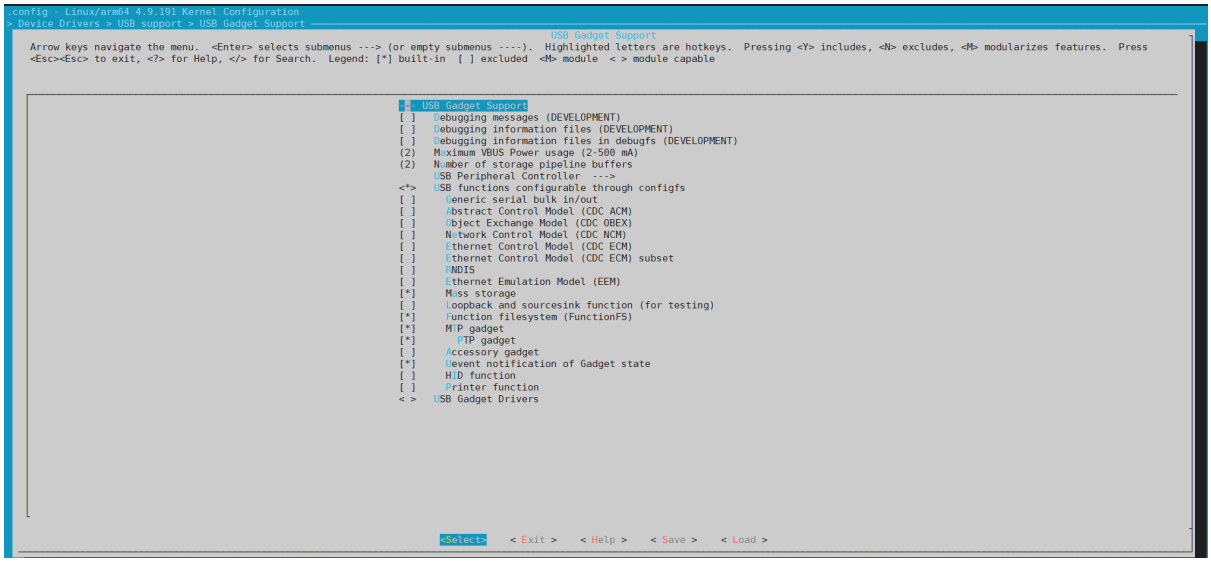

图 2-6: USB Gadget Support 详细配置

- 进入 USB Peripheral Controller，并打开下图选项：

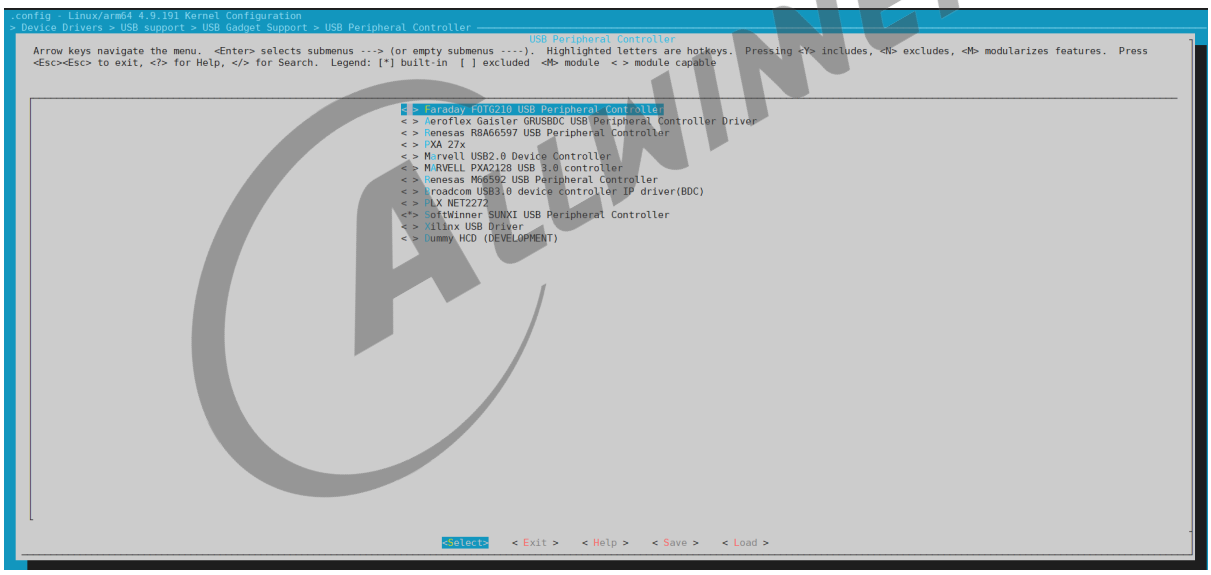

图 2-7: USB Peripheral Controller 详细配置

- 返回上一级，即 USB support，进入 SUNXI USB2.0 Dual Role controller support，并打开下图选项，如下图所示。

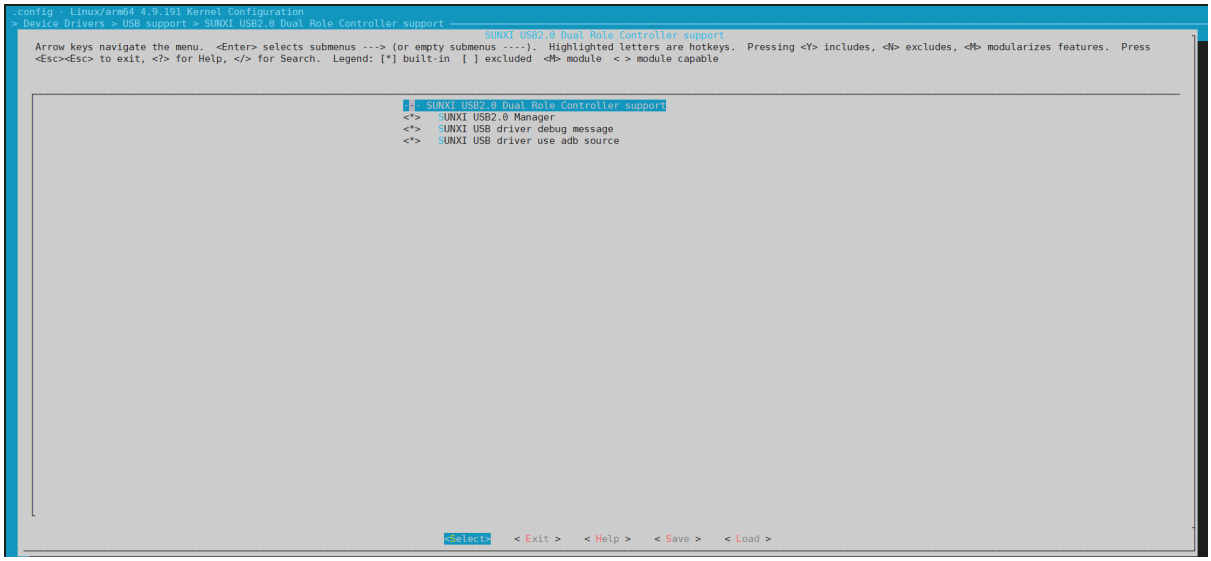

图 2-8: SUNXI USB2.0 Dual Role Controller Support 详细配置

- 若需支持 MTP PTP 等功能需开启 TYPEC 配置，返回上一级，即 USB support，进入 USB Type-C Support，并打开下图选项，如下图所示：

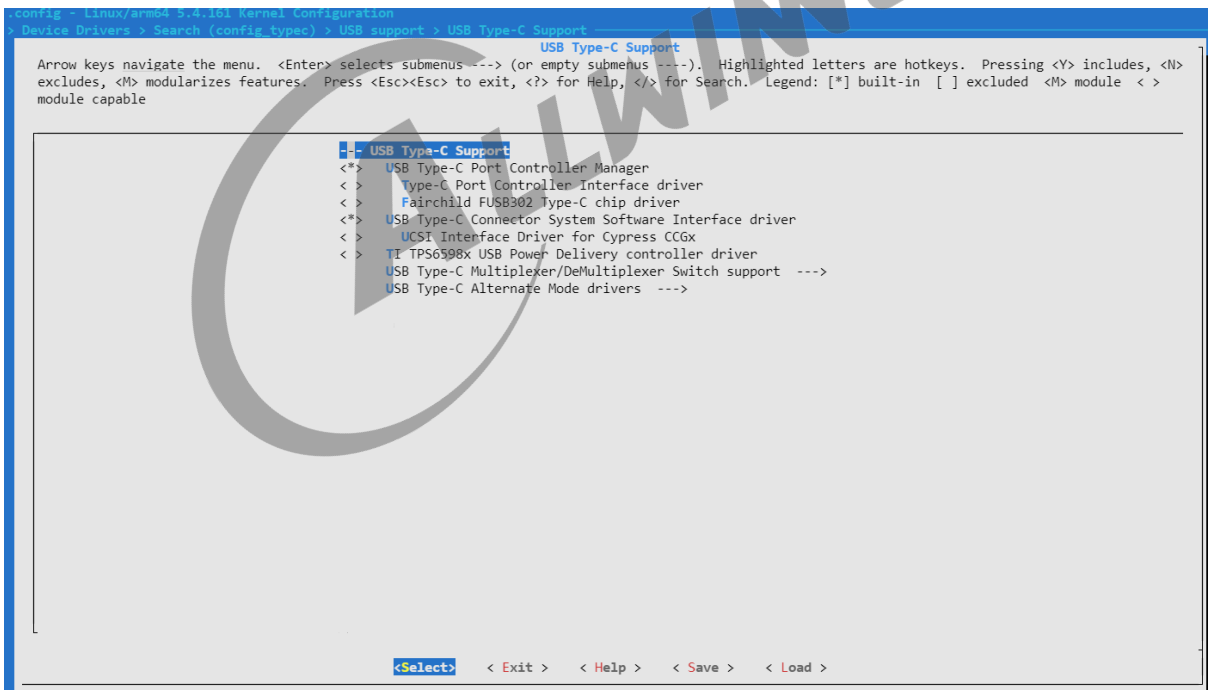

图 2-9: USB Type-C Support 详细配置

2.5 源码结构介绍

USB 驱动的源代码位于如下：

> linux-5.10 之前：drivers/usb
> linux-5.10 及之后：bsp/drivers/usb

如下是 sunxi 平台相关源码：

- Host

```
drivers/usb/host/
├── ehci_sunxi.c
├── ohci_sunxi.c
├── sunxi_hci.c
├── sunxi_hci.h
```

- UDC 和 Manager

```
drivers/usb/sunxi_usb/
├── include
│   ├── sunxi_hcd.h
│   ├── sunxi_sys_reg.h
│   ├── sunxi_udc.h
│   ├── sunxi_usb_board.h
│   ├── sunxi_usb_bsp.h
│   ├── sunxi_usb_config.h
│   ├── sunxi_usb_debug.h
│   └── sunxi_usb_typedef.h
├── Kconfig
├── Makefile
├── manager
│   ├── usbc0_platform.c
│   ├── usbc_platform.h
│   ├── usb_hcd_servers.c
│   ├── usb_hcd_servers.h
│   ├── usb_hw_scan.c
│   ├── usb_hw_scan.h
│   ├── usb_manager.c
│   ├── usb_manager.h
│   ├── usb_msg_center.c
│   └── usb_msg_center.h
├── misc
│   └── sunxi_usb_debug.c
├── udc
│   ├── sunxi_udc_board.c
│   ├── sunxi_udc_board.h
│   ├── sunxi_udc.c
│   ├── sunxi_udc_config.h
│   ├── sunxi_udc_debug.c
│   ├── sunxi_udc_debug.h
│   ├── sunxi_udc_dma.c
│   └── sunxi_udc_dma.h
└── usbc
    ├── usbc.c
    ├── usbc_dev.c
```

```
├── usbc_i.h
└── usbc_phy.c
```

2.6 驱动框架介绍

Linux 内核提供了完整的 USB 驱动程序框架。USB 总线采用树形结构，在一条总线上只能有唯一的主机设备。Linux 内核从主机和设备两个角度观察 USB 总线结构。下图是 Linux 内核从主机和设备两个角度观察 USB 总线结构的示意图。

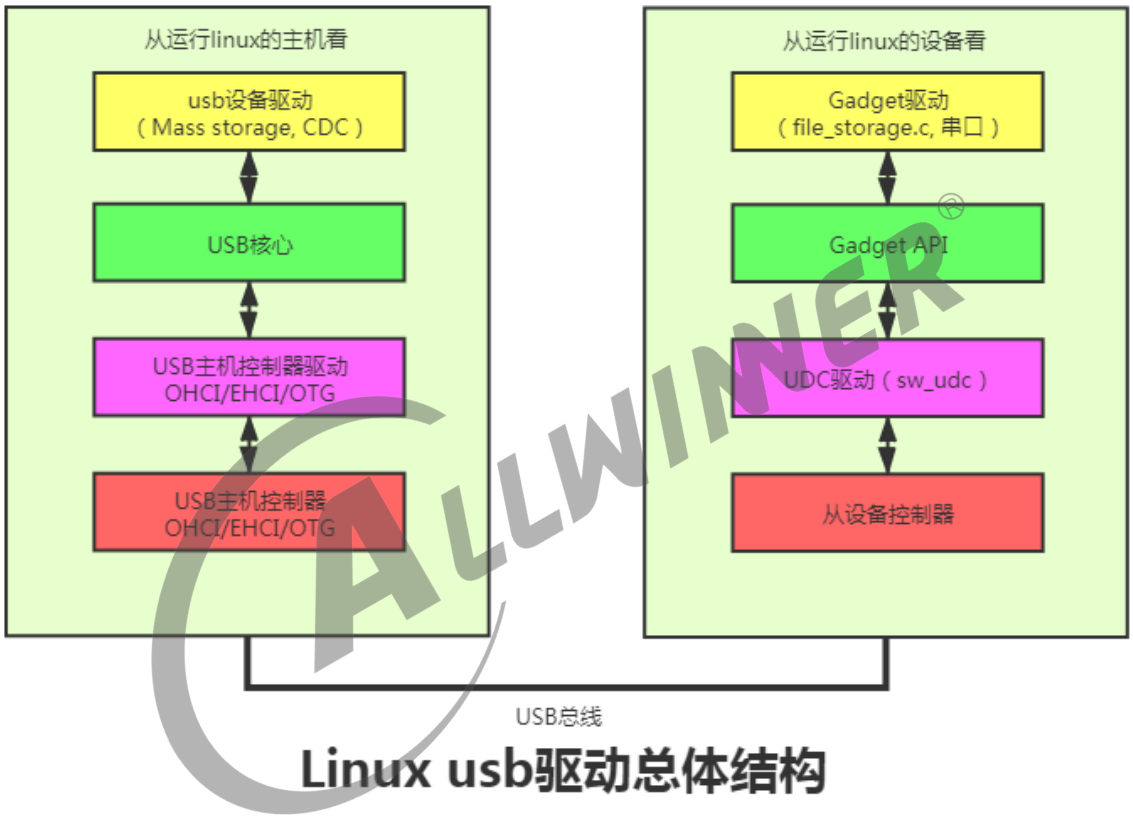

图 2-10: USB 驱动总体结构

USB 子系统主要任务包括：
a. 注册和管理设备驱动；
b. USB 设备寻找驱动，并初始化和配置设备；
c. 内核中表现设备的树形结构；
d. 与设备交互。

3 OTG 配置与使用

OTG（On-The-Go）通过 “ID” 确定设备，使得接口既能充当 HOST，亦能充当 DEVICE。在 AW IC 设计中，默认 USB0 为 OTG 接口。

3.1 OTG 检测原理

OTG 功能主要通过检测 ID 以及 det_vbus 的高低来确认加载 host 驱动还是 device 驱动。其中，根据版型的不同，检测方式也可以个性化定制，可以通过 dts 中的 usb_port_type、usb_detect_type、usb_detect_mode、usb_id_gpio、usb_det_vbus_gpio 来配置，具体参考**模块配置释义**小节。

OTG 设备识别是根据 VBUS 和 ID 的状态来决定的。对应的 usb0 的状态功能如下表所示：

表 3-1: OTG 组合检测状态

ID	VBUS	vbus_id_state	role
0	0	0x00	**otg-host**
0	1	0x01	**otg-host**
1	0	0x02	**otg-null**
1	1	0x03	**otg-device**

3.2 OTG 功能实现

OTG 功能的实现源码位于如下，OTG 功能是依赖于 manager 驱动来实现的。

```
/drivers/usb/sunxi_usb/manager
```

关于 manager 驱动的具体流程如下图所示：

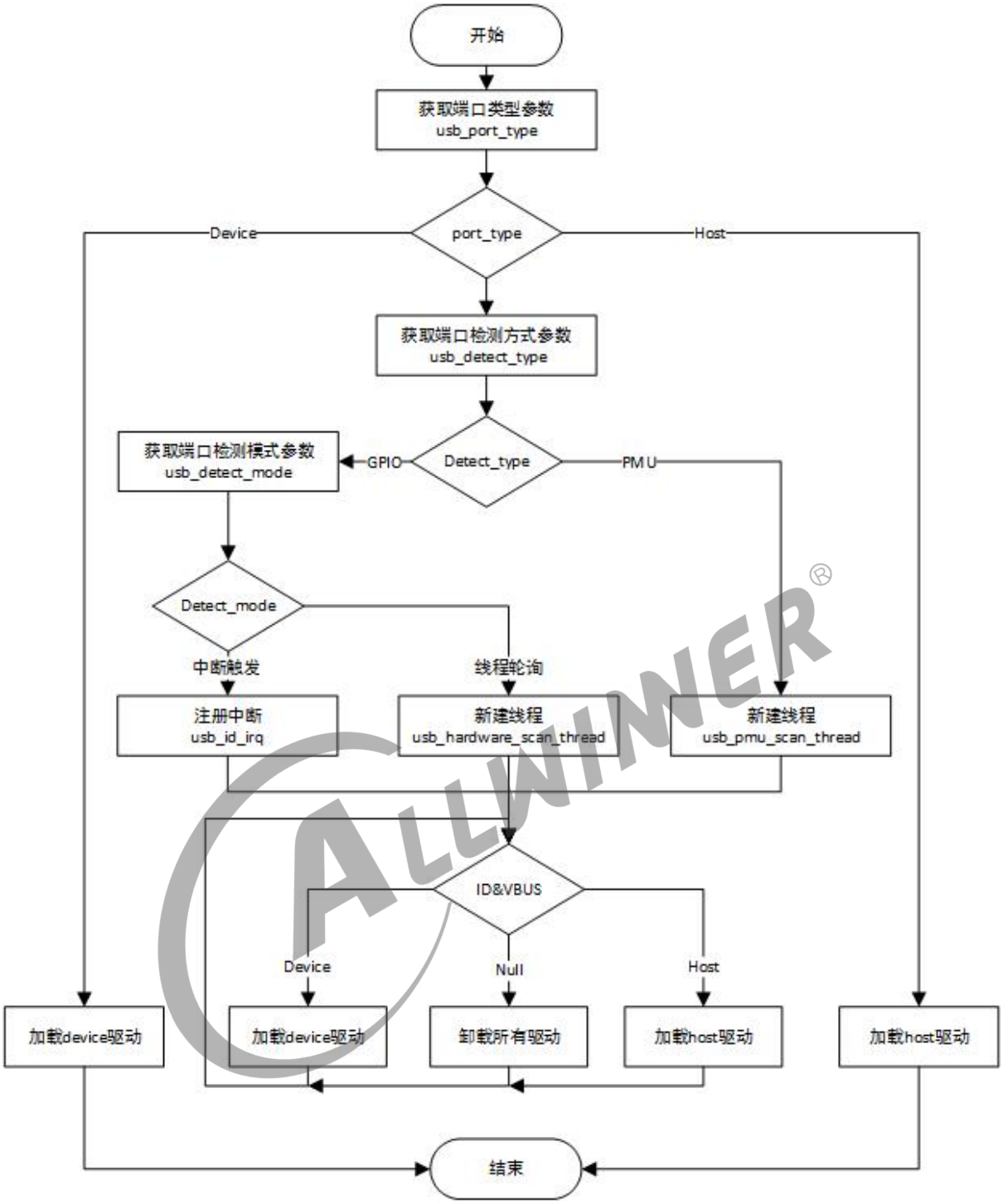

图 3-1: manager 实现流程图

4 KO 配置与使用

Linux 内核模块分为两种形态，一种是静态编译到内核，一种是通过 insmod 动态加载，通过在 menuconfig 中配置就能够实现。按照 **kernel menuconfig 配置说明**小节中的配置方式（配置为 Y），就是配置为静态编译。本章节主要介绍动态加载，即将模块配置为 M，再通过 insmod 对应的 ko 文件来实现加载。

4.1 模块 KO 配置

下面以 linux5.4 内核为例，介绍 USB 模块 ko 化配置。

步骤 1：选择板型

```
./build.sh config
```

步骤 2：打开内核配置菜单界面，将 USB 模块相关项配置为 M（能改尽改），注意 CONFIG_USB_GADGET 以及 CONFIG_USB_CONFIGFS 要保持 Y(其他模块对此存在依赖)

```
./build.sh menuconfig
```

修改后的设置项参考如下：

```
CONFIG_USB=y
CONFIG_USB_EHCI_HCD=m
CONFIG_USB_EHCI_HCD_SUNXI=m
CONFIG_USB_OHCI_HCD=m
CONFIG_USB_OHCI_HCD_SUNXI=m
CONFIG_USB_SUNXI_HCD=m
CONFIG_USB_SUNXI_HCI=m
CONFIG_USB_SUNXI_EHCI0=m
CONFIG_USB_SUNXI_EHCI1=m
CONFIG_USB_SUNXI_EHCI2=m
CONFIG_USB_SUNXI_OHCI0=m
CONFIG_USB_SUNXI_OHCI1=m
CONFIG_USB_SUNXI_OHCI2=m
CONFIG_USB_STORAGE=m
CONFIG_USB_STORAGE_REALTEK=m
CONFIG_USB_STORAGE_DATAFAB=m
CONFIG_USB_STORAGE_FREECOM=m
CONFIG_USB_STORAGE_ISD200=m
CONFIG_USB_STORAGE_USBAT=m
CONFIG_USB_STORAGE_SDDR09=m
CONFIG_USB_STORAGE_SDDR55=m
```

```
CONFIG_USB_STORAGE_JUMPSHOT=m
CONFIG_USB_STORAGE_ALAUDA=m
CONFIG_USB_STORAGE_KARMA=m
CONFIG_USB_STORAGE_CYPRESS_ATACB=m
CONFIG_USB_STORAGE_ENE_UB6250=m
CONFIG_USB_UAS=m
CONFIG_USB_GADGET=y
CONFIG_USB_SUNXI_UDC0=m
CONFIG_USB_CONFIGFS=y
CONFIG_USB_CONFIGFS_UEVENT=y
CONFIG_USB_CONFIGFS_MASS_STORAGE=y
CONFIG_USB_CONFIGFS_F_FS=y
CONFIG_USB_SUNXI_USB=m
CONFIG_USB_SUNXI_USB_MANAGER=m
CONFIG_USB_SUNXI_USB_DEBUG=m
CONFIG_USB_SUNXI_USB_ADB=m
CONFIG_USB_ROLE_SWITCH=m
```

步骤 3：将修改好的配置保存到配置文件中

```
./build.sh saveconfig
```

4.2 模块 KO 使用

步骤 1：编译、烧录、启动，进入如下目录（不同版本文件路径可能会有轻微差异）

```
/lib/modules/5.4.120+
```

步骤 2：手动 insmod 以下 ko 文件 (顺序有先后要求)

```
// 公共部分:
insmod usb-common.ko
insmod usbcore.ko
insmod ehci-hcd.ko

// 基础模块
insmod sunxi-hci.ko
insmod ehci-hcd.ko
insmod ehci-sunxi.ko
insmod ohci-hcd.ko
insmod ohci-sunxi.ko
insmod sunxi_usb_udc.ko
insmod sunxi_usbc.ko

// U盘功能
insmod uas.ko
insmod usb-storage.ko
insmod ums-realtek.ko
insmod ums-datafab.ko
insmod ums-freecom.ko
insmod ums-isd200.ko
insmod ums-usbat.ko
```

```
insmod ums-sddr09.ko
insmod ums-sddr55.ko
insmod ums-jumpshot.ko
insmod ums-alauda.ko
insmod ums-karma.ko
insmod ums-cypress.ko
insmod ums-eneub6250.ko
```

5 Gadget 配置与使用

Gadget 是指具有 USB 设备控制器的 USB 设备，根据具体的功能配置，连接到 PC 后可以作为 mass storage、uac 等设备。Linux 有原生 gadget 框架，通用的配置流程可参考下文。

5.1 Gadget 内核配置

需在 “USB functions configurable through configfs” 下选择需要的功能。

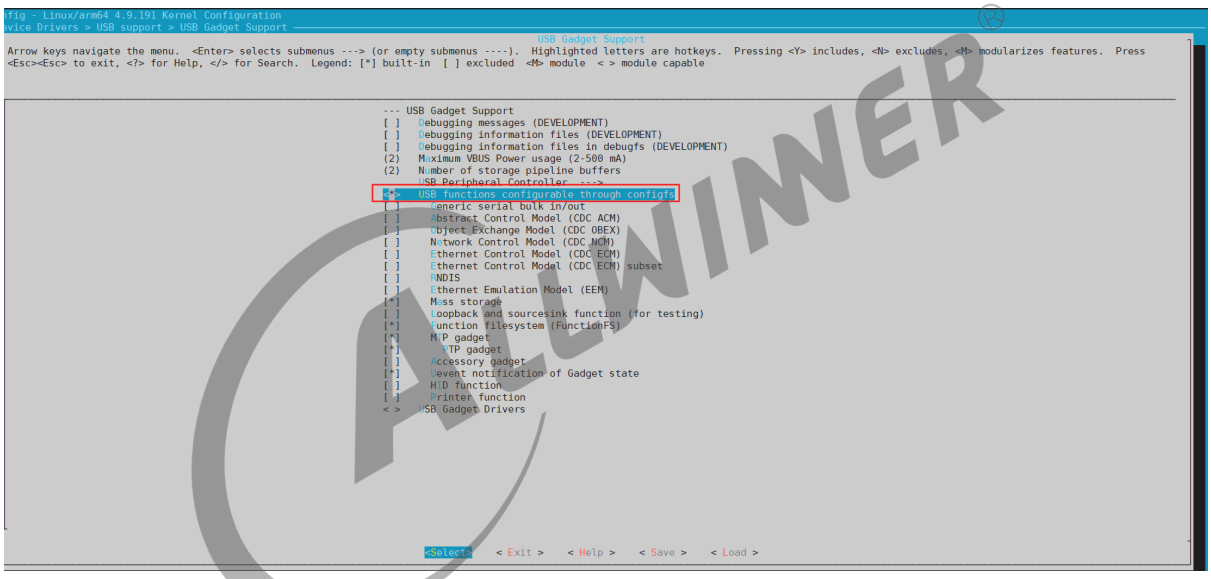

图 5-1: usb gadget 配置选择

5.2 Gadget 配置流程

Linux 使用 configfs 框架实现 composite gadget 功能。具体流程如下：

- 挂载 configs：

```
mount -t configfs none /sys/kernel/config
```

挂载完成之后在/sys/kernel/config 目录下就会生成 usb_gadget/目录。

- 建立 gadgets：

```
mkdir /sys/kernel/config/usb_gadget/g1
```

创建g1/目录之后，该目录下会生成很多配置目录，这里的g1表示 gadget 1，一个 UDC 对应一个 gadget，如果你的 SOC 上有多个 gadget，可以创建多个gx目录。

- 写入 gadget 的 PID、VID、序列号等信息：

```
echo "VID" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "PID" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
echo "manufacturer" > /sys/kernel/config/usb_gadget/g1/strings/0x409/manufacturer
echo "product" > /sys/kernel/config/usb_gadget/g1/strings/0x409/product
```

- 建立 gadget 相关配置 configurations

```
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
```

- 建立功能 functions

```
mkdir /sys/kernel/config/usb_gadget/g1/functions/<name>.<instance name>
```

：function name： 任意字符串

- 建立功能和配置的链接

```
ln -s /sys/kernel/config/usb_gadget/g1/functions/<name>.<instance name> /sys/kernel/config/usb_gadget/g1/configs/c
      .1
```

- 使能 gadget

```
echo <udc name> > UDC
```

常见 Gadget 功能的配置方式见附录。

5.3 Gadget 功能切换

如果系统中已经配置了某个 gadget 功能，此时要配置另一个功能，需要先将上一功能清理干净。

AW 系统中默认启动执行 adb 配置脚本，因此这里以清理 adb 功能为例进行说明。

```
//停止守护进程
./etc/adb_conf.sh stop

//清理相关功能链接
rm -fr /sys/kernel/config/usb_gadget/g1/configs/c.1/ffs.adb

//取消挂载
umount  /sys/kernel/config
```

执行以上操作，正常关闭 adb 后，根据需要的 gadget 功能，参考【附录】章节进行配置即可。

6 端点配置与使用

在 Gadget 配置使用过程中，可能出现端点的默认配置方式无法满足需求的情况，故需对端点进行修改满足需求。可参考现有的端点进行修改。譬如将批量端点改成中断端点，参考现有的中断端点进行修改即可。改动内容包括端点 fifo 大小，端点属性，端点方向。

6.1 分配端点 fifo

通过 drivers/usb/sunxi_usb/include/sunxi_udc.h 文件下的 ep_fifo 结构体数组来定义端点的 fifo 分配。

- name：端点名称
- fifo_addr：端点在 fifo 中的起始地址，通过上一端点的 fifo_addr+fifo_size 得出
- fifo_size：端点分配的 fifo 大小，一般与协议中规定的传输类型长度一致
- double_fifo：是否启动双 fifo 功能，若启用，fifo_size 需要填写 double_fifo 分配的长度

```
以8k平台为例：
static const struct sw_udc_fifo ep_fifo[] = {
    /*name,     fifo_addr,  fifo_size,  double_fifo*/
    {ep0name,     0,        512,        0},
    {ep1in_bulk_name, 512,      1024,       1},
    {ep1out_bulk_name, 1536,    1024,       1},
    {ep2in_bulk_name, 2560,     1024,       1},
    {ep2out_bulk_name, 3584,    1024,       1},
    {ep3_iso_name,    4608,     1024,       0},
    {ep4_int_name,    5632,     512,        0},
    {ep5in_bulk_name, 6144,     1024,       1},
    {ep5out_bulk_name, 7168,    1024,       1},
};
```

6.2 定义端点方向

通过 drivers/usb/sunxi_usb/include/sunxi_udc.h 文件下的 ep_fifo_in、ep_fifo_out 结构体数组来定义端点的 fifo 分配。

修改了 ep_fifo 的 name 属性后（如果修改了 in/out 或控制类型），则需要将对应的 in 端点和 out 端点分类更新，相应改动如下。

```
/**
 * ep_fifo_in[i] = {n} i: the physic ep index, n: ep_fifo's index for the ep
 * eg: ep_fifo_in[2] = {3} ===> ep2_in is in ep_fifo[3]
 * ep3_iso_name and ep4_int_name cannot be tx or rx simultaneously.
 */
static const int ep_fifo_in[] = {0, 1, 3, 5, 6, 7};
static const int ep_fifo_out[] = {0, 2, 4, 5, 6, 8};
};
```

说明

查看 ep_fifo 表来确认方向，参考流程如下：

1）ep0 为控制传输，单独使用端点 0 传输，in/out 均占用，端点 0 同时加入 ep_fifo_in/ep_fifo_out；

2）ep1 为批量传输，占用 1/2 端点分别 in/out，端点 1 加入 ep_fifo_in，端点 2 加入 ep_fifo_out；

3）ep2 为批量传输，占用 3/4 端点分别 in/out，端点 3 加入 ep_fifo_in，端点 4 加入 ep_fifo_out；

4）ep3 为同步传输，占用 5 端点 in/out，端点 5 同时加入 ep_fifo_in/ep_fifo_out；

5）ep4 为中断传输，占用 6 端点 in/out，端点 6 同时加入 ep_fifo_in/ep_fifo_out；

6）ep5 为批量传输，占用 7/8 端点分别 in/out，端点 7 加入 ep_fifo_in，端点 8 加入 ep_fifo_out。

6.3 设置端点属性

通过 drivers/usb/sunxi_usb/udc/sunxi_udc.c 文件下的 sunxi_udc 结构体数组来配置端点，修改了上述的端点属性之后，也需要同步修改端点的配置。

```
.ep[2] = {
    .num            = 1,
    .ep = {
        .name       = ep1out_bulk_name,
        .ops        = &sunxi_udc_ep_ops,
        .maxpacket  = SW_UDC_EP_FIFO_SIZE,
        .maxpacket_limit = SW_UDC_EP_FIFO_SIZE,
        .caps       = USB_EP_CAPS(USB_EP_CAPS_TYPE_BULK,
                    USB_EP_CAPS_DIR_OUT),
    },
    .dev            = &sunxi_udc,
    .bEndpointAddress = (USB_DIR_OUT | 1),
    .bmAttributes    = USB_ENDPOINT_XFER_BULK,
},
```

7 ACIN 配置方法

对于 AXP2202 的方案，硬件设计上可能会支持 ACIN，原理图及相关详细说明参考《Linux_PMIC_开发指南》FAQ 部分的 “**ACIN 属性配置**” 章节。

USB 驱动需要配置 USB0-DRVVBUS-EN 引脚，在 usbc0 节点增加如下配置即可：

```
usb_drvvbus_en_gpio = <&pio PH 7 GPIO_ACTIVE_HIGH>;
enable-active-high;
```

- 不配置时打印如下信息:

```
get drvvbus-en is fail , 22
```

- 添加空属性 usb_drvvbus_en_gpio 时打印如下信息:

```
get drvvbus-en is fail , 84
```

- 加载 host 驱动获取 gpio 成功时无额外打印，gpio request 失败时打印如下信息:

```
hci:  request ohci0-controller gpio:231
```

8 BOOST 供电配置方法

对于 AXP2202 的方案的非 USB0 口，常常需要给外设提供 5V 供电，但 PMU 在充电状态不能开启 BOOST，因此需要通过外部 IO 检测是否有外设接入，从而动态使能 vcc-5v，也能有效降低功耗，常见的电路设计如下图。

步骤 1：在 usbc1 节点增加如下配置，其中 usb_id_gpio 的作用类似于 USB0 口的 ID pin，故沿用以往的叫法，此处则是指 KD-EINT pin 脚。

```
usb_id_gpio = <&r_pio PL 10 GPIO_ACTIVE_HIGH>;
enable-active-high;
```

步骤 2：在 ehci1 和 ohci1 节点增加如下配置：

```
vbusin-supply = <&reg_vmid>;
```

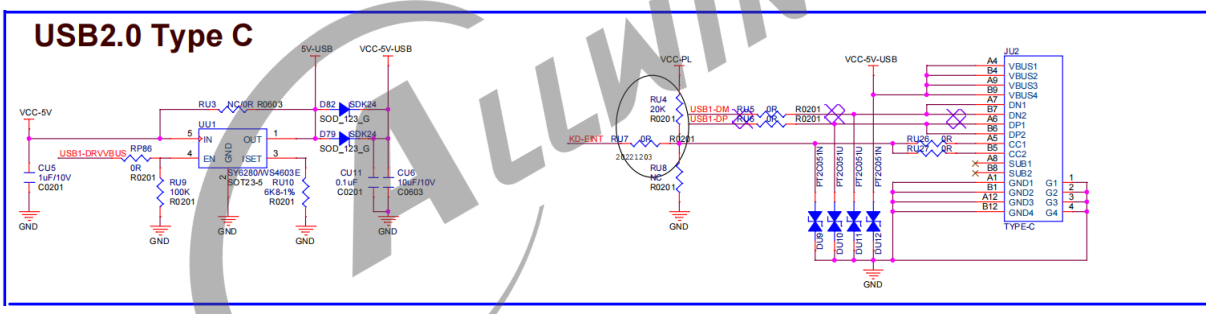

图 8-1: KD-EINT

9 USB Standby 调试方法

对于支持 **usb2 鼠标点击唤醒**的场景，在 5.4 及以前的内核版本通过直接修改 **hid-core.c** 的usbhid_start函数注册鼠标的唤醒源进行支持，在 5.10 以后的内核版本，为符合 Android 的 GKI 要求，不能够直接修改内核源码。

目前的应对策略为：

- Android 场景：通过 VendorHook 进行支持，详细实现可参考 **sunxi-vendor-hooks.c** 的sunxi_usb_new_device_added接口，其主要原理为在非 RootHub 设备接入时遍历其接口描述符，若确认为鼠标则调用device_init_wakeup重新初始化唤醒源。
- Linux 场景：对于 Tina 等，若无 GKI 要求，则可以通过直接修改内核源码进行支持，修改方法如下：

```
diff  -- git  a/drivers/hid/usbhid/hid-core.c b/drivers/hid/usbhid/hid-core.c
index be4c731aaa65..d3a6755cca09 100644
---  a/drivers/hid/usbhid/hid-core.c
+++ b/drivers/hid/usbhid/hid-core.c
@@ -1189,6 +1189,14 @@ static int usbhid_start(struct hid_device *hid)
                        device_set_wakeup_enable(&dev->dev, 1);
        }

+       /**
+        * NOTE: enable remote wakeup by default for all mouse devices
+        * supporting the boot protocol.
+        */
+       if  (interface->desc.bInterfaceSubClass == USB_INTERFACE_SUBCLASS_BOOT &&
+            interface->desc.bInterfaceProtocol == USB_INTERFACE_PROTOCOL_MOUSE)
+                device_set_wakeup_enable(&dev->dev, 1);
+
        mutex_unlock(&usbhid->mutex);
        return 0;

diff  -- git  a/drivers/hid/usbhid/usbmouse.c b/drivers/hid/usbhid/usbmouse.c
index 3fd93c2e4f4a..2fbc3f49e420 100644
---  a/drivers/hid/usbhid/usbmouse.c
+++ b/drivers/hid/usbhid/usbmouse.c
@@ -188,6 +188,7 @@ static int usb_mouse_probe(struct usb_interface *intf, const struct usb_device_i
                        goto fail3 ;

        usb_set_intfdata(intf,  mouse);
+       device_set_wakeup_enable(&dev->dev, 1);
        return 0;

 fail3 :
```

当然，也可以通过 SYSFS 下的调试节点手动使能鼠标的唤醒源，操作方法如下：

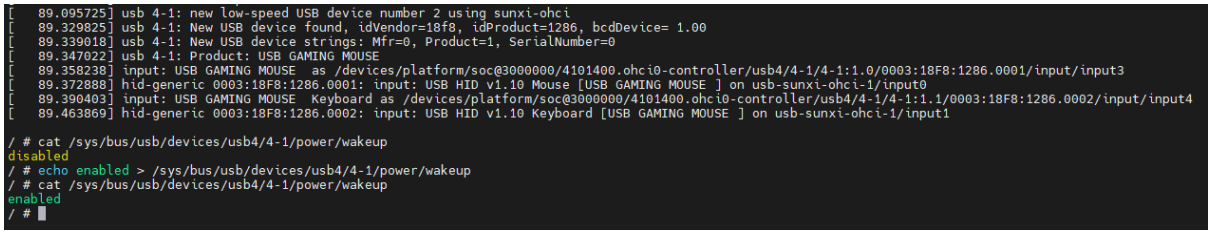

图 9-1: 修改节点支持鼠标休眠唤醒

测试指令为：（注，其路径与设备枚举时的具体路径有关）

```
cat /sys/bus/usb/devices/usb4/4-1/power/wakeup
echo enabled > /sys/bus/usb/devices/usb4/4-1/power/wakeup
```

10 调试方法

10.1 调试节点

usb 的调试节点一般位于/sys/devices/platform/soc/目录下，因型号差异会略有不同，可通过搜索找到对应的节点，参考命令如下：

```
find /sys/devices/ -name “节点名”
```

10.1.1 OTG 调试节点

- 查看当前 otg 的 role 角色：

```
cat otg_role
```

- 手动切换到 Device 模式

```
方法 1：cat usb_device
方法 2：echo usb_device > otg_role
```

- 手动切换到 Host 模式：

```
方法 1：cat usb_host
方法 2：echo usb_host > otg_role
```

- 手动切换到 Null 模式：

```
方法 1：cat usb_null
方法 2：echo null > otg_role
```

- 重新启动 OTG 检测：

```
cat usb_otg
```

- 开启 manager 的 debug 功能：

```
echo 1 > hw_scan_debug
```

说明

注意：手动切换 otg 之后，USB0 会关闭自动主从切换功能。开启 debug 也不会有信息打印。

10.1.2 HCI 调试节点

- 卸载主机驱动（可复位控制器）

```
echo 0 > ehci_enable
echo 0 > ohci_enable
```

- 加载主机驱动

```
echo 1 > ehci_enable
echo 1 > ohci_enable
```

10.1.3 DEVICE 调试

- 手动设置 dma debug 调试功能：

```
打开：echo 1 > dma_debug
关闭：echo 0 > dma_debug
```

- 手动设置中断 debug 调试功能：

```
打开：echo 1 > irq_debug
关闭：echo 0 > irq_debug
```

- 手动设置请求队列 debug 调试功能：

```
打开：echo 1 > queue_debug
关闭：echo 0 > queue_debug
```

- 手动设置 read fifo debug 调试功能：

```
打开：echo 1 > read_debug
关闭：echo 0 > read_debug
```

- 手动设置 write fifo debug 调试功能：

```
打开：echo 1 > write_debug
关闭：echo 0 > write_debug
```

- 手动设置 dma 传输功能（linux-5.4 及以上版本支持）：

```
打开：echo 1 > dma_enable
关闭：echo 0 > dma_enable
```

说明

注意 1：以上节点除 dma_enable 默认开启，其余均默认关闭。
注意 2：拔插 usb 时触发 reset 中断将会自动关闭dma_debug、irq_debug和queue_debug功能。
注意 3：所有节点均可通过 “cat 节点名” 的方式查看当前值。

10.2 读写寄存器

10.2.1 通过 sunxi_dump 读写寄存器

开启CONFIG_AW_DUMP_REG配置后，可以通过小机端/sys/class/sunxi_dump 下的节点打印寄存器的值，一般需要通过 Spec 知道寄存器的地址，简单用法如下：

- 进入节点所在目录

```
cd /sys/class/sunxi_dump/
```

- 查看单个寄存器值（0x410 为例）

```
echo 0x04100410 > dump; cat dump
```

- 修改单个寄存器值（0x410 为例）

```
echo 0x04100410 0x30 > write; cat write
```

- 查看批量寄存器值（以 ehci 为例）

```
echo 0x04200000,0x04200050 > dump; cat dump
```

10.2.2 通过 usb node 读写寄存器

USB 驱动加载过程中会初始化相关节点，用于查看寄存器，但未提供修改寄存器的功能。

- 查看 udc 寄存器值

```
cd /sys/class/udc_reg
cat udc_reg
```

说明

注意：当 device 驱动未加载时使用此节点会引起空指针死机。device 仅提供 udc_reg 用于查看寄存器。

- 查看 hci 寄存器值

```
cd /sys/class/hci_reg
查看 hci 寄存器：cat all_reg
查看 ehci 寄存器：cat ehci_reg
查看 ohci 寄存器：cat ohci_reg
查看 phy 寄存器：cat phy_reg
```

- 查看指定接口寄存器值

```
echo 0 > phy_reg
echo 1 > phy_reg
```

10.3 眼图测试

10.3.1 USB Device 眼图测试

- 获取 otg_ed_test 的路径 path

```
find /sys/ -name otg_ed_test
```

- 测试眼图命令

```
echo test_pack > path/otg_ed_test
```

10.3.2 USB Host 眼图测试

- 获取 ed_test 的路径 path

```
find /sys/ -name ed_test
```

- 测试眼图命令

```
echo test_pack > path/ed_test
```

10.4 驱动能力调节

- 寻找 phy_range 节点；

```
find ./ -name phy_range
```

- 根据寻找到的路径，读取异常 usb 口对应的 phy_range 值；

```
cat /sys/devices/xxx/…/phy_range
```

- 调节 phy_range 值，不同版型的含义不同，具体修改值需请教硬件相关同事

```
echo XXXX > /sys/devices/xxx/…/phy_range
```

- 确认 phy_range 值修改成功

```
cat /sys/devices/xxx/…/phy_range
```

11 日志分析

本章节主要提供 kernel 阶段初始化时相关的日志，以供分析和参考。不同的平台和不同版本的 kernel 之间可能会有轻微差异，但是大致流程是不变的。

在 Linux 系统中，USB 驱动分为 Host 端和 Device 端，两个角度所使用到的驱动是不一样的，具体可以参考 **驱动框架介绍**小节。

以下按照初始化的时间顺序来展示相关的 log。

11.1 USB Host 日志分析

- 注册 USB 文件系统，生成/sys/bus/usb/目录。

```
[0.328384] usbcore: registered new interface driver usbfs
```

- 注册 USB HUB 驱动。

```
[0.334047] usbcore: registered new interface driver hub
```

- 注册 USB HUB 通用设备驱动，即 usb_generic_driver。通常 USB 设备都是以设备的身份先与 usb_generic_driver 匹配，匹配成功后再生成对应接口，并调用 device_add() ，从而完成接口与接口驱动的匹配。

```
[0.339454] usbcore: registered new device driver usb
```

- 注册设备类驱动，具体项可以通过 menuconfig 配置。

```
[1.316453] usbcore: registered new interface driver uas
[1.316644] usbcore: registered new interface driver usb-storage
[1.316769] usbcore: registered new interface driver ums-alauda
[1.316894] usbcore: registered new interface driver ums-cypress
[1.317030] usbcore: registered new interface driver ums-datafab
[1.317158] usbcore: registered new interface driver ums_eneub6250
[1.317292] usbcore: registered new interface driver ums-freecom
[1.317429] usbcore: registered new interface driver ums-isd200
```

```
[1.317564] usbcore: registered new interface driver ums-jumpshot
[1.317686] usbcore: registered new interface driver ums-karma
[1.317814] usbcore: registered new interface driver ums-onetouch
[1.317964] usbcore: registered new interface driver ums-realtek
[1.318095] usbcore: registered new interface driver ums-sddr09
[1.318220] usbcore: registered new interface driver ums-sddr55
[1.318356] usbcore: registered new interface driver ums-usbat
[1.643335] usbcore: registered new interface driver usbhid
[1.643337] usbhid: USB HID core driver
```

- 初始化 EHCI 控制器驱动。USB0 默认为 OTG 模式，因此在初始化的时候不加载驱动。其余的接口则逐个加载驱动。从 log 中可以获取控制器基本信息（中断号、设备虚拟地址、控制器版本等），比如，ehci1 控制器被枚举为一个 USB2.0 USB hub，分配了 bus number 1。该阶段由于各种因素影响，初始化时序各不相同，因此打印的时间点可能会有点不一样。

```
[1.384213] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[1.391500] sunxi-ehci: EHCI SUNXI driver
[2.799023] [sunxi-ehci0]: probe, pdev->name: 4101000.ehci0-controller, sunxi_ehci: 0xc153f528, 0x:f1215000, irq_no:4e
[2.810951] [sunxi-ehci0]: Not init ehci0
[2.815839] [sunxi-ehci1]: probe, pdev->name: 4200000.ehci1-controller, sunxi_ehci: 0xc153fa48, 0x:f1219000, irq_no:50
[2.841003] sunxi-ehci 4200000.ehci1-controller: EHCI Host Controller
[2.848197] sunxi-ehci 4200000.ehci1-controller: new USB bus registered, assigned bus number 1
[2.857992] sunxi-ehci 4200000.ehci1-controller: irq 80, io mem 0x04200000
[2.878599] sunxi-ehci 4200000.ehci1-controller: USB 2.0 started, EHCI 1.00
[2.886565] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002, bcdDevice= 5.04
[2.895783] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[2.903832] usb usb1: Product: EHCI Host Controller
[2.909267] usb usb1: Manufacturer: Linux 5.4.99 ehci_hcd
[2.915281] usb usb1: SerialNumber: sunxi-ehci
[2.920771] hub 1-0:1.0: USB hub found
[2.924980] hub 1-0:1.0: 1 port detected
```

- 初始化 OHCI 控制器驱动。同理初始化的时候不加载 USB0 驱动。该 ohci1 控制器被枚举为一个 USB1.1 USB hub，分配了 bus number 3。

```
[1.396360] ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
[1.403250] sunxi-ohci: OHCI SUNXI driver
[3.054393] [sunxi-ohci0]: probe, pdev->name: 4101400.ohci0-controller, sunxi_ohci: 0xc153f7b8
[3.064000] [sunxi-ohci0]: Not init ohci0
[3.068901] [sunxi-ohci1]: probe, pdev->name: 4200400.ohci1-controller, sunxi_ohci: 0xc153fcd8
[3.090625] sunxi-ohci 4200400.ohci1-controller: OHCI Host Controller
[3.097816] sunxi-ohci 4200400.ohci1-controller: new USB bus registered, assigned bus number 3
[3.107572] sunxi-ohci 4200400.ohci1-controller: irq 81, io mem 0x04200400
[3.178739] usb usb3: New USB device found, idVendor=1d6b, idProduct=0001, bcdDevice= 5.04
[3.187945] usb usb3: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[3.195991] usb usb3: Product: OHCI Host Controller
[3.201423] usb usb3: Manufacturer: Linux 5.4.99 ohci_hcd
[3.207437] usb usb3: SerialNumber: sunxi-ohci
[3.212916] hub 3-0:1.0: USB hub found
[3.217118] hub 3-0:1.0: 1 port detected
```

- 插入一个设备，例如：U 盘。

```
[6.078261] usb 1-1: new high-speed USB device number 2 using sunxi-ehci
[6.240654] usb 1-1: New USB device found, idVendor=048d, idProduct=1234, bcdDevice= 2.00
[6.249870] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[6.257946] usb 1-1: Product: Disk 3.0
[6.262173] usb 1-1: Manufacturer: USB
[6.266418] usb 1-1: SerialNumber: 4147281022783708332
[6.274430] usb-storage 1-1:1.0: USB Mass Storage device detected
[6.282739] scsi host0: usb-storage 1-1:1.0
[7.291439] scsi 0:0:0:0: Direct-Access     VendorCo ProductCode     2.00 PQ: 0 ANSI: 4
[7.303355] sd 0:0:0:0: [sda] 65536000 512-byte logical blocks: (33.6 GB/31.3 GiB)
[7.312854] sd 0:0:0:0: [sda] Write Protect is off
[7.318308] sd 0:0:0:0: [sda] Mode Sense: 03 00 00 00
[7.324848] sd 0:0:0:0: [sda] No Caching mode page found
[7.330899] sd 0:0:0:0: [sda] Assuming drive cache: write through
[7.383596]  sda: sda1
[7.402650] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

- 拔出一个设备，例如：U 盘。

```
[8.851554] usb 3-1: USB disconnect, device number 4
```

- 卸载主机驱动。通过 cat usb_null（USB0 用 usb_null）节点，可以卸载当前驱动。

```
[8.091190] [ehci1-controller]: sunxi_usb_disable_ehci
[8.097040] [sunxi-ehci1]: remove, pdev->name: 4200000.ehci1-controller, sunxi_ehci: 0xc153fa48
[8.109470] sunxi-ehci 4200000.ehci1-controller: remove, state 4
[8.118266] usb usb1: USB disconnect, device number 1
[8.126850] sunxi-ehci 4200000.ehci1-controller: USB bus 1 deregistered
```

11.2 USB Device 日志分析

- 初始化 Device 控制器驱动。

```
[5.497364] insmod_device_driver
[5.497364]
device_chose finished!
```

- 作为设备插入主机端，如：通过 usb 线连接到 PC（adb）。

```
[8.528955] android_work: sent uevent USB_STATE=CONNECTED
[8.561758] configfs-gadget gadget: high-speed config #1: b
[8.577329] android_work: sent uevent USB_STATE=CONFIGURED
```

- 作为设备拔出，如：拔掉连接到 PC 的 usb 线（adb）。

```
[7.511214] sunxi_vbus_det_work()3310 WARN: get power supply failed
[7.521785] android_work: sent uevent USB_STATE=DISCONNECTED
[7.567262] regulator-dummy: Underflow of regulator enable count
[7.574047] android_work: did not send uevent (0 0 00000000)
```

- 卸载设备驱动。通过 cat usb_null（或者拔掉 usb 线）节点，可以卸载当前驱动。

```
[8.268634] rmmod_device_driver
[8.268634]
[8.275559] rmmod_device_driver()223 WARN: get power supply failed
[8.282630] android_work: did not send uevent (0 0 00000000)
```

12 FAQ

12.1 USB 基本功能异常排查

12.1.1 USB Host 基本功能异常排查步骤

- 多找几个 USB 设备试试，排除个别 USB 设备本身的问题。
- 多更换几根 USB 线缆试试，排除个别 USB 线缆的问题。
- 多找几个 PC 主机做相同的实验，作为参考对比。若在 PC 有相同现象，则认为正常。
- 若硬件有多个 USB 口，尝试同样条件下测试其他 USB 口的主机功能是否正常。
- 样机设备 USB 口外接独立供电的 USB-HUB 设备，再将 USB 设备连接到 USB-HUB 上，确认主机功能是否正常。
- 确认主机驱动是否加载成功。

  （1）若为 USB0 口，则可通过如下方式确认:

  ```
  cat /sys/devices/platform/soc/usbc0/otg_role
  ```

  （2）若为 USB1 口，可通过如下方式确认:

  ```
  cat sys/devices/platform/soc/5200000.ehci1-controller/ehci_enable
  cat sys/devices/platform/soc/5200000.ohci1-controller/ohci_enable
  若为0，则没有加载Host驱动。
  ```

- 重新加载 Host 驱动，确认此时功能是否正常。

  （1）若为 USB0 口，则可通过如下方式:

  ```
  方式1：重新插拔OTG线。
  方式2：手动切换到Host模式。
  ```

  （2）若为 USB1 口，则可通过卸载驱动、再加载驱动。
- 对比 SDK 代码与最新发布的代码或者补丁, 确认代码是否更新到最新。
- 同样条件下，分别打印出功能异常板子和功能正常板子的相关寄存器，并进行对比，确认是否有不同之处。
- 同样条件下，使用 USB 分析仪抓取正常及异常时候的数据包，，并进行对比，确认是否有不同之处。
- 出现异常时，测试 USB 高速眼图是否正常。若眼图测试未通过，可尝试调节眼图参数。

12.1.2 USB Device 基本功能异常排查步骤

- 多换几个 PC 主机做相同的测试，排除个别 PC 的问题。
- 多更换几根 USB 线缆做相同的测试，排除个别 USB 线缆的问题。
- 确认 Device 驱动是否加载成功，可通过如下方式:

  （1）通过 Log。

```
[ 104.732695] insmod_device_driver
[ 104.732695]
device_chose finished!
```

  （2）通过节点查看当前 Role。

- 重新加载 Device 驱动，确认此时功能是否恢复正常。

  （1）重新插拔 USB 线。

  （2）手动切换到 Device 模式。

- 对比 SDK 代码与最新发布的代码或者补丁, 确认代码是否更新到最新。
- 同样条件下，分别打印出功能异常板子和功能正常板子的相关寄存器，并进行对比，确认是否有异常。
- 同样条件下，使用 USB 分析仪抓取正常及异常时候的数据包，，并进行对比，确认是否有不同之处。
- 出现异常时，确认 USB 高速眼图是否正常。

12.2 USB 常见问题思路分析

12.2.1 插拔设备后无法识别

样机设备 USB 口插入 USB 设备后，日志中未看到有效信息。该问题分两种情况，常规情况插入所有的设备均无反应和概率性情况排查插入个别设备无反应，概率性情况从步骤三开始排查

步骤 1，如果是 usb0 口，检查对应方案的 sys_config 配置是否正确；

```
;---        USB0控制标志
;-------------------------------
[usbc0]
usbc0_used            = 1
usb_port_type         = 2     0 : device ;  1 : host ;  2 : otg
usb_detect_type       = 1
usb_detect_mode       = 0
usb_id_gpio           = port:PH7<0><1><default><default>
usb_det_vbus_gpio     = "axp_ctrl"
usb_drv_vbus_gpio     = "axp_ctrl"
usb_host_init_state   = 0     若默认为host，此处需要置1
usb_wakeup_suspend    = 0
;---        USB Device
usb_luns              = 3
usb_serial_unique     = 0
usb_serial_number     = "20080411"
```

图 12-1: usb0 口 sys_config 配置

步骤 2，如果是 usb0 口，确认当前是否为 host 模式；

```
# cat ‘busybox find /sys/devices/ -name “otg_role”‘
```

若为 host，则 cat 结果为：usb_host，若非 host 模式，则可以通过下述命令切换:

```
# cat ‘busybox find /sys/devices/ -name “usb_host”‘
（关键信息）
[67823.051097] insmod_host_driver
[67823.051097]
[67823.057952] [ehci0-controller]: sunxi_usb_enable_ehci
[67823.064383] [sunxi-ehci0]: probe, pdev->name: 5101000.ehci0-controller, sunxi_ehci: 0xc10b0e14, 0x:f08fb000, irq_no:cb
[67823.076901] sunxi-ehci 5101000.ehci0-controller: SW USB2.0 ‘Enhanced’ Host Controller (EHCI) Driver
[67823.087856] sunxi-ehci 5101000.ehci0-controller: new USB bus registered, assigned bus number 1
[67823.098241] sunxi-ehci 5101000.ehci0-controller: irq 203, io mem 0xc014c1f8
[67823.125814] sunxi-ehci 5101000.ehci0-controller: USB 0.0 started, EHCI 1.00
[67823.134458] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[67823.142255] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[67823.151283] usb usb1: Product: SW USB2.0 ‘Enhanced’ Host Controller (EHCI) Driver
[67823.160367] usb usb1: Manufacturer: Linux 4.9.118 ehci_hcd
[67823.166845] usb usb1: SerialNumber: sunxi-ehci
[67823.173215] hub 1-0:1.0: USB hub found
```

```
[67823.177592] hub 1-0:1.0: 1 port detected
[67823.182663] [ohci0-controller]: sunxi_usb_enable_ohci
[67823.188386] [sunxi-ohci0]: probe, pdev->name: 5101000.ohci0-controller, sunxi_ohci: 0xc10b05c4
[67823.198349] sunxi-ohci 5101000.ohci0-controller: SW USB2.0 'Open' Host Controller (OHCI) Driver
[67823.208321] sunxi-ohci 5101000.ohci0-controller: new USB bus registered, assigned bus number 2
[67823.218255] sunxi-ohci 5101000.ohci0-controller: irq 204, io mem 0xef10fd54
[67823.288067] usb usb2: New USB device found, idVendor=1d6b, idProduct=0001
[67823.295836] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[67823.304140] usb usb2: Product: SW USB2.0 'Open' Host Controller (OHCI) Driver
[67823.312173] usb usb2: Manufacturer: Linux 4.9.118 ohci_hcd
[67823.318493] usb usb2: SerialNumber: sunxi-ohci
[67823.325126] hub 2-0:1.0: USB hub found
[67823.329636] hub 2-0:1.0: 1 port detected
host_chose finished!
```

步骤 3，样机设备 USB 口外接独立供电的 USB-HUB 设备，将 USB 设备连接到 USB-HUB 上，检查是否正常初始化和挂载。

步骤 4，如果经过步骤 3 仍然无法正常识别，请按照 USB Host 基本功能异常排查步骤进一步排查。

12.2.2 插入设备后反复重连

样机设备 USB 口插入 USB 设备后，循环打印 connect、disconnect，即表现为如下 log

```
[11:1615:02:01:483]ehci_irq: highspeed device connect
[11:1615:02:01:491]ehci_irq: highspeed device disconnect
[11:1615:02:01:512]ehci_irq: highspeed device connect
[11:1615:02:01:513]ehci_irq: highspeed device disconnect
[11:1615:02:01:532]ehci_irq: highspeed device connect
[11:1615:02:01:533]ehci_irq: highspeed device disconnect
```

请按照 USB Host 基本功能异常排查步骤进行排查，重点排除设备、信号、眼图原因。

12.2.3 高速设备异常切换 OHCI

样机设备 USB 口插入 USB 高速设备后，ehci 对设备枚举异常，切换到使用 ohci。即表现为如下 log。

```
[  89.260115] usb 1-1: new high-speed USB device number 33 using sunxi-ehci
[  89.400152] usb 1-1: device descriptor read/64, error -71
[  89.640231] usb 1-1: device descriptor read/64, error -71
[  90.350260] usb 2-1: new full-speed USB device number 14 using sunxi-ohci
[  90.563274] usb 2-1: not running at top speed; connect to a high speed hub
[  90.582287] usb 2-1: New USB device found, idVendor=12d1, idProduct=107e
```

请按照 USB Host 基本功能异常排查步骤进行排查，重点排除设备、信号、眼图原因。

12.2.4 插入设备再启动不识别

样机的 USB 口插入 USB 设备开机，不识别设备。

- 步骤 1，若为 usb0 口，根据模块配置介绍检查确认 usb 的配置；
- 步骤 2，确认启动后插盘是否可以识别——排查开机时序；
- 步骤 3，确认插盘启动不识别是否是个别 usb 设备行为——排查设备兼容性；
- 步骤 4，若当前使用的是 usb0 口，确认其他 usb 口插盘启动是否可正常识别——排查接口差异；
- 步骤 5，分别就启动后和再次切到 host 模式后两种场景，依次打印 0x0~0x54，0x400~0x454，0x800~0x830 寄存器，对比是否有异常，基本操作方法：

```
echo 0x5101800,0x5101830 > /sys/class/sunxi_dump/dump
cat /sys/class/sunxi_dump/dump
```

- 步骤 4，如果经过步骤 3 没确定异常原因，请按照 USB Host 基本功能异常排查步骤进行排查。

说明

基地址请参考对应的 SPEC。

12.2.5 休眠唤醒后设备无法识别

系统进入 super standby 休眠，唤醒后，USB 设备无法识别。

- 步骤 1，确认 vbus 是否可控，需保证设备是在休眠中断电，唤醒时上电的前提下进行测试。
- 步骤 2，卸载主机驱动、再重新加载；查看设备是否可以正常识别（需根据具体路径，以下仅供参考）

```
echo 0 > /sys/devices/platform/soc/5101000.ehci0-controller/ehci_enable
echo 1 > /sys/devices/platform/soc/5101000.ehci0-controller/ehci_enable
```

- 步骤 3，用协议分析仪抓取异常时的包，跟正常对比。
- 步骤 4，若上述无发现，请根据 USB Host 基本功能异常排查步骤进行排查。

12.2.6 休眠唤醒后设备枚举失败

插着设备的情况下，系统进入 super standby 休眠，唤醒过程中拔掉设备，唤醒后进行插拔设备测试，会概率性出现枚举不上，如无法识别设备、降速使用 ohci 等。

- 步骤 1，确认 vbus 是否可控，需保证设备是在休眠中断电，唤醒时上电的前提下进行测试。
- 步骤 2，延长唤醒时开 vbus 的时间（同步修改 sunxi_ehci_resume_work 和 sunxi_ohci_resume_work），测试能否复现；

```
static void sunxi_ehci_resume_work(struct work_struct *work)
{
    struct sunxi_hci_hcd *sunxi_ehci = NULL;

    sunxi_ehci = container_of(work, struct sunxi_hci_hcd, resume_work);

    /* Waiting hci to resume. */
    msleep(5000);

    sunxi_hcd_board_set_vbus(sunxi_ehci, 1);
    atomic_sub(1, &g_sunxi_usb_super_standby);
}
```

图 12-2: 延长唤醒时开 vbus 的时间

- 步骤 3，异常出现后，卸载主机驱动、再重新加载，再进行设备插拔，测试能否复现；

```
echo 0 > /sys/devices/soc/1c14000.ehci0-controller/ehci_enable
echo 0 > /sys/devices/soc/1c14000.ohci0-controller/ohci_enable


echo 1 > /sys/devices/soc/1c14000.ehci0-controller/ehci_enable
echo 1 > /sys/devices/soc/1c14000.ohci0-controller/ohci_enable
```

- 步骤 4，用示波器分别抓取开机、唤醒阶段 VCC_USB、VDD_SYS，VBUS 的上电时序，并测试电压值，对比唤醒过程中上述三路电压是否有异常。

- 步骤 5，测试异常时候的 usb 眼图，确认是否有异常。
- 步骤 6，若上述无发现，请按照 USB Host 基本功能异常排查步骤进行排查。

12.2.7 使用 MTP 互传数据卡死

PC 和设备之间，使用 MTP 协议传输数据时概率性卡住，比如非常多文件、超大文件的时候。

- 步骤 1，确定 mtp 设备被 PC 正常枚举，且驱动加载正确。
- 步骤 2，同样测试条件下，用协议分析仪分别对正常传输和异常传输的情况抓包，并对比。
- 步骤 3，mtp 设备被 PC 正确识别后，分别打印正常和异常设备的寄存器，并对比。
- 步骤 4，若上述无发现，请按照 USB Device 基本功能异常进行排查。

12.2.8 Camera 设备帧率低

在 H3(linux-3.4) 上，usb 摄像头达不到 720p@15 规格，但是在 H6(linux-3.10) 上，该摄像头可以达到该规格，怀疑带宽不足引起该问题。

- 步骤 1，确认在 H3 上的现象，是单个摄像头的个性问题还是所有摄像头共性的问题。
- 步骤 2，抓包确认复现问题时采用的格式。
- 步骤 3，根据抓包的枚举信息，确认步骤 2 中采用的格式是否支持规格 720p@15。

12.2.9 UAC 数据丢失

使用 UAC 播放 pcm 格式的 wav 文件，dump 的数据和源文件匹配不上，怀疑同步传输丢包。

- 步骤 1，用协议分析仪抓包，确认数据传输的端点（默认端点 3 为 ISO）。
- 步骤 2，增加 debug 信息，确认该数据传输端点对应的中断数和整个传输过程中 fifo 中收到的总数据量；对比是否跟抓包数据相匹配。
- 步骤 3，将 udc 中断绑定到其他 cpu，再次确认步骤 2 中的数据是否相匹配。
- 步骤 4，禁止掉串口打印；再次确认步骤 2 中的数据是否相匹配。

12.2.10 USB3.0 转千兆网口吞吐量低

对 USB3.0 转千兆网口进行吞吐量测试，测试吞吐量数据只有 82.9 Mbits/sec。

- 步骤 1，使用移动硬盘对 usb3.0 口进行读写速率测试，确认 usb 主控的实际速写性能。
- 步骤 2，更换网卡驱动，确认吞吐量是否有改善。
- 步骤 3，将 usb 0x05200800 寄存器从 0x701 修改为 0x1。

12.2.11 插入 OTG U 盘（3.0）无法识别

对于标准的 Type-C 接口，通过 USB3.0 的 OTG 线连接 USB3.0 的外设，有一侧总是无法正常识别（仅针对 DPDM 引到 USB2.0 控制器，而不连接 3.0 控制器的情况）。

- 方法 1：需要调试 GMA340 分离器件以支持 USB3 接口 Type-C 线的正反插功能。
- 方法 2：在仅开启 USB2.0 控制器驱动的场景需要将 GMA340 配置为关断状态，也能够正常使用 USB2.0 的控制器进行 USB3.0 外设的正反插。

  在 usbc0 节点增加如下配置即可：

```
usb_gma340_oe_gpio = <&pio PH 8 GPIO_ACTIVE_HIGH>;
enable-active-high;
```

13 附录

13.1 Gadget 配置示例

13.1.1 小机做 mass storage

依赖配置项：

```
CONFIG_USB_CONFIGFS_MASS_STORAGE=y

Device Drivers  --->
 [*] USB support  --->
  <*>  USB Gadget Support  --->
   <*>  USB functions configurable through configfs
    [*]    Mass storage
```

configfs 配置方法：

```
dd if=/dev/zero of=/dev/a.bin bs=1M count=100
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x18d1" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0001" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0
echo /dev/a.bin > /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/lun.0/file
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

说明

如果需要增加 lun，在 functions/mass_storage.usb0 下：
mkdir lun.1
mkdir lun.2

13.1.2 小机做 cdrom

依赖配置项：

```
CONFIG_USB_CONFIGFS_MASS_STORAGE=y

Device Drivers  --->
 [*] USB support  --->
  <*>  USB Gadget Support  --->
   <*>  USB functions configurable through configfs
    [*]    Mass storage
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1f3a" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0xa4ac" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0
echo 1 > /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/lun.0/cdrom
echo /tmp/phoenixcard.iso > /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/lun.0/file
ln -s /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
      mass_storage.usb0
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

说明

/tmp/phoenixcard.iso 根据实际情况更改（任意文件修改后缀即可）。

13.1.3 小机做 UAC1

依赖配置项：

```
CONFIG_SOUND=y
CONFIG_SND=y
CONFIG_USB_CONFIGFS_F_UAC1=y

Device Drivers  --->
 <*> Sound card support  --->
  <*>  Advanced Linux Sound Architecture  --->
 [*] USB support  --->
  <*>  USB Gadget Support  --->
   <*>  USB functions configurable through configfs
    [*]    Audio Class 1.0
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1d61" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0101" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/uac1.usb0
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
```

```
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/uac1.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

注意：默认代码 UDC 的 endpoint 中仅有一个 ISO 端点，如果出现以下报错，可通过修改 udc 的端点配置修复（将 int 端点配置为 iso）。

```
[ 4342.289380] ERR: sunxi_udc_dequeue: driver is null
[ 4342.294854] configfs-gadget 4100000.udc-controller: failed to start g1: -19
echo: write error: No such device
```

13.1.4 小机做 UAC2

依赖配置项：

```
CONFIG_SOUND=y
CONFIG_SND=y
CONFIG_USB_CONFIGFS_F_UAC2=y

Device Drivers  --->
 <*> Sound card support  --->
  <*>  Advanced Linux Sound Architecture  --->
 [*] USB support  --->
  <*>  USB Gadget Support  --->
   <*>  USB functions configurable through configfs
    [*]    Audio Class 2.0
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1d61" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0101" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/uac2.usb0
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/uac2.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

13.1.5 小机做 UVC

依赖配置项：

```
CONFIG_MEDIA_CAMERA_SUPPORT=y
CONFIG_USB_CONFIGFS_F_UVC=y

Device Drivers  --->
 <*> Multimedia support  --->
  [*]   Cameras/video grabbers support
 [*] USB support  --->
  <*>   USB Gadget Support  --->
    <*>   USB functions configurable through configfs
     [*]    Audio Class 2.0
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1f3a" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x100d" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0
mkdir -p /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p
echo 1280 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/wWidth
echo 720 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/wHeight
echo 333333 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/dwFrameInterval
echo 333333 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/
        dwDefaultFrameInterval
echo 442368000 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/dwMinBitRate
echo 442368000 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/dwMaxBitRate
echo 1843200 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/
        dwMaxVideoFrameBufferSize
mkdir /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/header/h
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/ /sys/kernel/config/usb_gadget/g1/
        functions/uvc.usb0/streaming/header/h/
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/header/h/ /sys/kernel/config/usb_gadget/g1/
        functions/uvc.usb0/streaming/class/fs
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/header/h/ /sys/kernel/config/usb_gadget/g1/
        functions/uvc.usb0/streaming/class/hs
mkdir /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/control/header/h
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/control/header/h/ /sys/kernel/config/usb_gadget/g1/
        functions/uvc.usb0/control/class/fs/
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/control/header/h/ /sys/kernel/config/usb_gadget/g1/
        functions/uvc.usb0/control/class/ss/
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

注意：如果测试 UAC 时将 int 端点配置为 iso 端点会由于申请不到 int 端点出现以下报错。

```
[  218.744777] configfs-gadget gadget: uvc: uvc_function_bind()
[  218.751185] configfs-gadget gadget: uvc: Unable to allocate control EP
[  218.758534] ERR: sunxi_udc_dequeue: driver is null
[  218.763923] configfs-gadget 4100000.udc-controller: failed to start g1: -22
echo: write error: Invalid argument
```

13.1.6 小机做 HID

依赖配置项：

```
CONFIG_USB_CONFIGFS_F_HID=y

Device Drivers  --->
 [*] USB support  --->
  <*>  USB Gadget Support  --->
   <*>  USB functions configurable through configfs
    [*]    HID function
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config/
mkdir /sys/kernel/config/usb_gadget/g1
echo 0x0525 >/sys/kernel/config/usb_gadget/g1/idVendor
echo 0xa4ac >/sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/hid.usb0
echo 512 >/sys/kernel/config/usb_gadget/g1/functions/hid.usb0/report_length
echo -ne <report_desc> >/sys/kernel/config/usb_gadget/g1/functions/hid.usb0/report_desc
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 >/sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 >/sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/hid.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

说明

report_desc 根据需求自定义，示例如下。

```
\\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\\x15\\x00\\x25\\x01\\x75\\x01\\x95\\x08\\x81\\x02\\x95\\
      x01\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01\\x05\\x08\\x19\\x01\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\
      x03\\x95\\x06\\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\\x81\\x00\\xc0
```

注意：默认代码 UDC 的 endpoint 中仅有一个 INT 端点，如果出现以下报错，可通过修改 udc 的端点配置修复（将 iso 端点配置为 int）。

```
[  272.667636] configfs-gadget gadget: hidg_bind FAILED
[  272.673219] ERR: sunxi_udc_dequeue: driver is null
[  272.678634] configfs-gadget 4100000.udc-controller: failed to start g1: -19
echo: write error: No such device
```

13.1.7 小机做 rndis

依赖配置项：

```
CONFIG_USB_CONFIGFS_RNDIS=y

Device Drivers  --->
```

```
[*] USB support  --->
  <*>  USB Gadget Support  --->
    <*>  USB functions configurable through configfs
      [*]    RNDIS
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1f3a" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x200a" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/rndis.usb0
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/rndis.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/rndis.usb0
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

13.1.8 小机做 acm

依赖配置项：

```
CONFIG_USB_ACM=y
CONFIG_USB_CONFIGFS_ACM=y

Device Drivers  --->
 [*] USB support  --->
   <*>  USB Modem (CDC ACM) support
   <*>  USB Gadget Support  --->
     <*>  USB functions configurable through configfs
       [*]    Abstract Control Model (CDC ACM)
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1f3a" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0007" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/acm.usb0
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/acm.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/acm.usb0
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

注意：出现以下报错时，检查 CONFIG_USB_ACM 配置是否打开。

```
[  87.430781] Config c/1 of g1 needs at least one function.
[  87.436844] ERR: sunxi_udc_dequeue: driver is null
[  87.442269] configfs-gadget 4100000.udc-controller: failed to start g1: -22
echo: write error: Invalid argument
```

13.1.9 小机做 adb

依赖配置项：

```
CONFIG_NET=y
CONFIG_UNIX=y
CONFIG_INET=y
CONFIG_USB_CONFIGFS_F_FS=y

[*] Networking support  --->
 Networking options  --->
  <*> Unix domain sockets
  [*] TCP/IP networking
Device Drivers  --->
 [*] USB support  --->
  <*>  USB Gadget Support  --->
   <*>  USB functions configurable through configfs
    [*]    Function filesystem (FunctionFS)
```

configfs 配置方法：

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x18d1" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0002" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
echo "20080411" > /sys/kernel/config/usb_gadget/g1/strings/0x409/serialnumber
echo "Android" > /sys/kernel/config/usb_gadget/g1/strings/0x409/manufacturer
mkdir /sys/kernel/config/usb_gadget/g1/functions/ffs.adb
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/ffs.adb/ /sys/kernel/config/usb_gadget/g1/configs/c.1/ffs.adb
mkdir /dev/usb-ffs
mkdir /dev/usb-ffs/adb
mount -o uid=2000,gid=2000 -t functionfs adb /dev/usb-ffs/adb/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

注意：按照上述配置出现以下报错时需要执行adbd &开启 adb 守护进程。

```
[  69.369687] ERR: sunxi_udc_dequeue: driver is null
[  69.375097] configfs-gadget 4100000.udc-controller: failed to start g1: -19
echo: write error: No such device
```

13.1.10 小机做 mass storage+adb

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x18d1" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0003" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
echo "20080411" > /sys/kernel/config/usb_gadget/g1/strings/0x409/serialnumber
echo "Android" > /sys/kernel/config/usb_gadget/g1/strings/0x409/manufacturer
mkdir /sys/kernel/config/usb_gadget/g1/functions/ffs.adb
mkdir /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0
echo ${BLOCK_PATH} > /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/lun.0/file
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
      mass_storage.usb0
ln -s /sys/kernel/config/usb_gadget/g1/functions/ffs.adb/ /sys/kernel/config/usb_gadget/g1/configs/c.1/ffs.adb
mkdir /dev/usb-ffs
mkdir /dev/usb-ffs/adb
mount -o uid=2000,gid=2000 -t functionfs adb /dev/usb-ffs/adb/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

13.1.11 小机做 uvc+uac1

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1f3a" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x100d" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/uac1.usb0
mkdir /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0
mkdir -p /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p
echo 1280 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/wWidth
echo 720 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/wHeight
echo 333333 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/dwFrameInterval
echo 333333 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/
      dwDefaultFrameInterval
echo 442368000 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/dwMinBitRate
echo 442368000 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/dwMaxBitRate
echo 1843200 > /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/720p/
      dwMaxVideoFrameBufferSize
mkdir /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/header/h
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/mjpeg/m/ /sys/kernel/config/usb_gadget/g1/
      functions/uvc.usb0/streaming/header/h/
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/header/h/ /sys/kernel/config/usb_gadget/g1/
      functions/uvc.usb0/streaming/class/fs
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/streaming/header/h/ /sys/kernel/config/usb_gadget/g1/
      functions/uvc.usb0/streaming/class/hs
mkdir /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/control/header/h
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/control/header/h/ /sys/kernel/config/usb_gadget/g1/
      functions/uvc.usb0/control/class/fs/
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/control/header/h/ /sys/kernel/config/usb_gadget/g1/
      functions/uvc.usb0/control/class/ss/
```

```
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/uvc.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ln -s /sys/kernel/config/usb_gadget/g1/functions/uac1.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

13.1.12 小机做 hid+cdrom

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x1f3a" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0xa4ac" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 > /sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 > /sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/hid.usb0
mkdir /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0
echo 512 >/sys/kernel/config/usb_gadget/g1/functions/hid.usb0/report_length
echo -ne <report_desc> >/sys/kernel/config/usb_gadget/g1/functions/hid.usb0/report_desc
ln -s /sys/kernel/config/usb_gadget/g1/functions/hid.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/hid.usb0
echo 1 > /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/lun.0/cdrom
echo /tmp/phoenixcard.iso > /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/lun.0/file
ln -s /sys/kernel/config/usb_gadget/g1/functions/mass_storage.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/
      mass_storage.usb0
ls /sys/class/udc/ | xargs echo > /sys/kernel/config/usb_gadget/g1/UDC
```

13.1.13 小机做 rndis+adb

```
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/g1
echo "0x18d1" > /sys/kernel/config/usb_gadget/g1/idVendor
echo "0x0010" > /sys/kernel/config/usb_gadget/g1/idProduct
mkdir /sys/kernel/config/usb_gadget/g1/strings/0x409
mkdir /sys/kernel/config/usb_gadget/g1/functions/ffs.adb
mkdir /sys/kernel/config/usb_gadget/g1/functions/rndis.usb0
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1
echo 0xc0 >/sys/kernel/config/usb_gadget/g1/configs/c.1/bmAttributes
echo 500 >/sys/kernel/config/usb_gadget/g1/configs/c.1/MaxPower
mkdir /sys/kernel/config/usb_gadget/g1/configs/c.1/strings/0x409
ln -s /sys/kernel/config/usb_gadget/g1/functions/rndis.usb0/ /sys/kernel/config/usb_gadget/g1/configs/c.1/rndis.usb0
ln -s /sys/kernel/config/usb_gadget/g1/functions/ffs.adb/ /sys/kernel/config/usb_gadget/g1/configs/c.1/ffs.adb
mkdir /dev/usb-ffs
mkdir /dev/usb-ffs/adb
mount -o uid=2000,gid=2000 -t functionfs adb /dev/usb-ffs/adb/
ls/sys/class/udc/|xargs echo>/sys/kernel/config/usb_gadget/g1/UDC
```

著作权声明

商标声明

免责声明