Rockchip
GMAC 开发指南

发布版本:1.0

日期:2017.02

前言

概述

本文可适用于所有使用 Rockchip 以太网功能的芯片。本文将针对 RMII/RGMII 接口芯片在 RK322XH SDK 上的配置做详细的描述。

产品版本

芯片名称	内核版本
RK322XH	3.10
RK3328	3.10

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

日期	版本	作者	修改说明
20117.02.17	V1.0	Wdc	加入 RMII/RGMII 功能

目录

1 以太网 MAC PHY 芯片

由于在 RK 系列的 SoC 中内置了以太网 MAC 控制器，所以只需要搭配一颗以太网 PHY 芯片，即可实现以太网卡功能。按照规范，即使是不同厂家的 PHY，仍然有一部分寄存器的定义是通用的，只要配置了这些通用的寄存器，基本上 PHY 就可以正常工作。因此，在 Linux 驱动中有通用的 PHY 驱动，目前的芯片所配套的 SDK 中使用的都是通用驱动，当然 SoC 中的 MAC 驱动是需要实现的。10/100M 以太网 PHY 与 MAC 之间的接口主要有 MII 和 RMII。 10/100/1000M 以太网 PHY 与 MAC 之间的接口主要有 RGMII。

2 GMAC 模块

以太网控制器在 kernel 中的 menuconfig 配置如下所示，可以根据产品需求开关此配置，SDK 中下述配置默认打开：

```
Device Drivers  --->
    Networking support  --->
        [*]  Ethernet (10 or 100Mbit)  --->
            RockChip devices --->
                RockChip 10/100/1000 Ethernet driver
```

GMAC 驱动代码位于 drivers/net/ethernet/rockchip/gmac/*。

2.1 GMAC 外接 PHY 的 DTS 配置详解

以下是 GMAC + 外置 PHY DTS 部分配置的详细说明：

```
&gmac_clkin {
    /* PHY 供给 GMAC 的时钟大小 */
    clock-frequency = <125000000>;
};

&gmac{
    /* 给 PHY 供电的 PMU 上的 ldo */
    pmu_regulator = "act_ldo5";
    /* ldo 有效电平 1->HIGH, 0->LOW */
    pmu_enable_level = <1>;
    /* 电源控制 IO 及有效电平 */
    power-gpio = <&gpio0 GPIO_A6 GPIO_ACTIVE_HIGH>;
    /* 复位 IO 及有效电平*/
    reset-gpio = <&gpio3 GPIO_B0 GPIO_ACTIVE_LOW>;
    /* PHY 接口 */
    phy-mode = "rgmii";
    /*时钟方向 */
    clock_in_out = "input";
    /* TX 线上的延时值 */
    tx_delay = <0x30>;
    /* RX 线上的延时值 */
    rx_delay = <0x10>;
};
```

一般情况下，控制供电方式或者为 PMU，或者为 IO 控制，若使用其中一种，则屏蔽另外一种。还有一种情况，是对 PHY 长供电，这时候可以考虑把二者都屏蔽。

PHY 接口的配置：根据板上所使用的 PHY 的接口进行配置，一般千兆 PHY 为 RGMII，百兆 PHY 为 RMII。

时钟方向： input 表示由 PHY 提供，output 表示由 GMAC 提供。

TX/RX 线上的延时值：tx 与 rx 的 delay skew;

PHY 供给 GMAC 的时钟大小：这个值只有在时钟方向为 input 时才会使用到，RMII 接口时，

设为 50000000, RGMII 接口时设为 125000000。

2.2 外置 100M PHY 配置

以下是 GMAC + 外置 100M PHY DTS 部分配置修改部分

```
&gmac_clkin {
    /* 如果使用外部时钟，修改成 50M */
    clock-frequency = <50000000>;
};

&gmac{
    power-gpio = <&gpio0 GPIO_A6 GPIO_ACTIVE_HIGH>;
    reset-gpio = <&gpio4 GPIO_B0 GPIO_ACTIVE_LOW>;
    /* 修改成 rmii */
    phy-mode = "rmii";
    /* 修改成 output，也就是由 RK 主控提供 */
    clock_in_out = "output";
    /* 百兆不需要配置延迟 */
    /* tx_delay = <0x30>; */
    /* rx_delay = <0x10>; */
};
```

2.3 外置 1000M PHY 配置

以下是 GMAC + 外置 1000M PHY DTS 部分配置修改部分

```
&gmac_clkin {
    /* 如果使用外部时钟，修改成 125M */
    clock-frequency = <125000000>;
};

&gmac{
    power-gpio = <&gpio0 GPIO_A6 GPIO_ACTIVE_HIGH>;
    reset-gpio = <&gpio4 GPIO_B0 GPIO_ACTIVE_LOW>;
    /* 修改成 rmii */
    phy-mode = "rmii";
    /* 修改成 output，也就是由 RK 主控提供 */
    clock_in_out = "output";
    /* 百兆不需要配置延迟 */
    /* tx_delay = <0x30>; */
    /* rx_delay = <0x10>; */
};
```

注意：125M 主时间需要由外部 PHY 提供，这是因为 RK 主控分不出 125M clock 或分出的 clock 可能不精准，会造成 GMAC 丢包或无法工作。

2.4 RK3328 GMAC DTS 配置

RK3228H/RK3328 带有两个 GMAC 控制器，并且内置一个 100M PHY，可作双网卡。一是 GMAC+外部 100M/1000M, dts 上表示为 gmac2io 节点，另一个是 GMAC+内部 100M PHY, dts 上表示为 gmac2phy 节点。

参考配置文件在 arch/arm64/boot/dts/rk322xh-evb.dtsi 网卡 0，需要外接 PHY 使用,：

```
&gmac_clkin {
        clock-frequency = <125000000>;
};

/* 千兆 phy 配置 */
&gmac2io {
        /* pmu_regulator = "act_ldo5"; */
        /* power-gpio = <&gpio0 GPIO_A6 GPIO_ACTIVE_HIGH>; */
        reset-gpio = <&gpio1 GPIO_C2 GPIO_ACTIVE_LOW>;
        /* phyirq-gpio = <&gpio0 GPIO_B1 GPIO_ACTIVE_LOW>; */
        phy-mode = "rgmii";
        pinctrl-names = "default";
        pinctrl-0 = <&rgmiim1_pins>;
        clock_in_out = "input";
        /* delay line for rgmii + 1000M mode */
        tx_delay = <0x26>;
        rx_delay = <0x11>;
        status = " okay";
};
```

网卡 1，使用内部 100M PHY:

```
&gmac2phy {
    clock_in_out = "output";
    pinctrl-names = "default";
    /* 网口 led 灯控制 */
    pinctrl-0 = <&fephyled_rxm1 &fephyled_linkm1>;
    status = "okay";
};
```

内部 phy 的 led 网口灯控制是可以改变的，只要改变 pinctrl 的配置即可，最多控制两个，配置成下面的两个功能 pin 就可以实现对相应功能的 led 灯控制。

```
gmac2phy {
        fephyled_speed100: fephyled-speed100 {
            rockchip,pins =
                <0 GPIO_D7 RK_FUNC_1 &pcfg_pull_none>;
        };

        fephyled_speed10: fephyled-speed10 {
            rockchip,pins =
                <0 GPIO_D6 RK_FUNC_1 &pcfg_pull_none>;
        };

        fephyled_duplex: fephyled-duplex {
            rockchip,pins =
                <0 GPIO_D6 RK_FUNC_2 &pcfg_pull_none>;
        };
```

```
            fephyled_rxm0: fephyled-rxm0 {
                rockchip,pins =
                    <0 GPIO_D5 RK_FUNC_1 &pcfg_pull_none>;
            };

            fephyled_txm0: fephyled-txm0 {
                rockchip,pins =
                    <0 GPIO_D5 RK_FUNC_2 &pcfg_pull_none>;
            };

            fephyled_linkm0: fephyled-linkm0 {
                rockchip,pins =
                    <0 GPIO_D4 RK_FUNC_1 &pcfg_pull_none>;
            };

            fephyled_rxm1: fephyled-rxm1 {
                rockchip,pins =
                    <2 GPIO_D1 RK_FUNC_2 &pcfg_pull_none>;
            };

            fephyled_txm1: fephyled-txm1 {
                rockchip,pins =
                    <2 GPIO_D1 RK_FUNC_3 &pcfg_pull_none>;
            };

            fephyled_linkm1: fephyled-linkm1 {
                rockchip,pins =
                    <2 GPIO_D0 RK_FUNC_2 &pcfg_pull_none>;
            };
        };
```

3 MAC 地址

通常，每个以太网设备只有唯一的 MAC 地址，所以需要有一个地方用来存储这个唯一的地址，同时在打开以太网时读取出这个地址，并写入 PHY 寄存器。SDK 提供了四种获取以太网 MAC 地址的方法：

1） 存储在 NAND 的 IDB 中

首先要保证 kernel 中的配置 CONFIG_ETH_MAC_FROM_IDB 已打开

其次要使用烧写工具将地址写入，烧写工具在 SDK 中有提供。

2） 存储在 EEPROM 中

首先要保证 kernel 中的配置 CONFIG_ETH_MAC_FROM_EEPROM 已打开

其次 EEPROM 的驱动见 drivers/staging/rk29/eeprom，根据不同型号请自行作相应修改

3） 使用 WiFi 的 MAC 地址

该种方法的原理是在系统启动时自动加载一次 Wi-Fi 驱动，同时将 Wi-Fi 的 MAC 地址读出并存储在/data 分区的一个文件中，以太网打开时，读取该文件中的地址。

首先要保证 kernel 中的配置 CONFIG_ETH_MAC_FROM_WIFI_MAC 已打开

其次要保证 Android 上 wlan_mac 程序存在，且已在 init.rc 或 init.rkxx.rc 中已添加如下脚本

```
service wlan_mac /system/bin/wlan_mac
    class main
    oneshot
```

以太网驱动读取地址的代码存于 drivers/net/eth_mac，请根据实际需求修改此代码。

由于不同的网络设备的 MAC 地址必须是唯一的，所以请考虑使用这种方法的风险性。

4） 使用随机地址

若上述三种方法均未采用，驱动中会在每次打开以太网时随机生成 MAC 地址,由于不同的网络设备的 MAC 地址必须是唯一的，所以请考虑使用这种方法的风险性。

4 GMAC 常见问题排查

4.1 DMA initialization failed

如果 kernel 打印以下异常 log：

```
stmmac_open: DMA initialization failed
```

这种情况只有在“clock_in_out = "input"情况下才出现。

A) 需要确认 GMAC 工作主时钟 MAC_CLK 是否有从 PHY 供给主控：

使用 100M PHY 时，其频率是 50M

使用 1000M PHY 时，其频率是 125M

B) 如果有 clock，需要确认 clock 的幅度是否达标，一般需要 3.0V 以上

C) 需要确认 iomux 是否正确

4.2 PHY 初始化异常

如果出现 PHY 初始化异常

类似如下异常打印：

```
stmmac_open: Cannot attach to PHY
```

或

```
eth0: No PHY found
```

PHY 正常识别会有类似如下打印：

```
eth0: PHY ID 20005c90 at 1 IRQ 0 (stmmac-0:01) active
```

A. 需要先确认硬件是否有异常，对比 RK 发布的以太网 PHY 参考电路,或者找 RK 硬件同事 check 下原理图

B. 需要确认 PHY 的供电是否正常，如 VCC_LAN 等电源脚

C. 如果 PHY 有 reset 脚控制，确认是否正常控制到

D. 测量 MAC 信号脚的电平，需要工作在 3.3V

E. 还可以尝试增加以下 delay 时间试试

```
+++ b/drivers/net/ethernet/rockchip/gmac/stmmac_platform.c
@@ -358,10 +358,10 @@ static intphy_power_on(bool enable)
                //reset
if (gpio_is_valid(bsp_priv->reset_io)) {
gpio_direction_output(bsp_priv->reset_io, bsp_priv->reset_io_level);
-                       mdelay(5);
+                        mdelay(100);
gpio_direction_output(bsp_priv->reset_io, !bsp_priv->reset_io_level);
                }
-               mdelay(30);
+                mdelay(100);
        } else {
                //pull down reset
```

4.3 GAMC 引脚复用 IOMUX 确认

GMAC RGMII 接口与其它功能脚复用（无法同时使用），需要确认 IOMUX 状态是否正确。例如,出现如下打印（可在 kernel log 中搜索 pinctrl）：

```
rk3368-pinctrl pinctrl.20: pin gpio3-8 already requested by ff680000.pwm; canno
t claim for ff290000.eth
```

通过上面 log 看到：发现 gpio3b0 先被 PWM 申请了，导致无法再被 GMAC 申请。可通过 io 命令直接查看 IOMUX 寄存器确认。

4.4 工作时钟确认

确认 GMAC 工作主时钟 MAC_CLK 是否正常.

如果用的是 100M PHY，MAC_CLK 的频率是 50M；如果用的是 1000M PHY，MAC_CLK 的频率是 125M.125M clock 必须由 PHY 提供，100M clock 可由 RK 主控提供。

例如 RTL8211E PHY 规格书中关于 125M clock 描述如下：

46	1	CLK125	O/PD	125MHz Reference Clock Generated from Internal PLL. This pin should be kept floating if the 125MHz clock is not be used for MAC.

1. 测量 MAC_CLK 引脚是否有 clock，频率是否正常
   1000M 时，如果 PHY 输出的 clock 不是 125M，需要检查 PHY 外围电路是否正常
   确认 iomux 是否正常
2. clock 的幅度是否达标，一般需要 3.0V 以上
3. 通过 clk_summary 确认

主控端可通过以下 clock tree 信息：cat d/clk/clk_summary 来确认 clock 是否设置正确(注意对应的 enable_cnt 需要为 1，这才表示这个 clock 已经使能)

例如由 PHY 提供 125M clock:

```
clockenable_cntprepare_cnt          rate
gmac_clkin                  1           1           125000000
```

例如由主控提供 50M clock:

```
clockenable_cntprepare_cnt          rate
clk_mac_pll                 1           1           50000000
```

注意：如果出现 4.2 PHY 初始化失败这一节中的情况，那么 clock 会被 disable 掉，所以看到的 clock 会是以下情况(对应的 enable_cnt 为 0)：

```
clockenable_cntprepare_cnt          rate
clk_mac_pll                 0           0           50000000
```

可加以下调试代码，异常时不去关 clock 来调试

```
+++ b/drivers/net/ethernet/rockchip/gmac/stmmac_platform.c
@@ -219,6 +219,7 @@ static intgmac_clk_enable(bool enable) {
structbsp_priv * bsp_priv = &g_bsp_priv;
phy_iface = bsp_priv->phy_iface;
+           enable = 1;
if (enable) {
if (!bsp_priv->clk_enable) {
```

4.5 无法接收 RX 数据

无法接收 RX 数据表现出的现象是：无法获取到 IP 地址，或者配置静态 IP 地址后无法上网。

正常情况下通过网线连接上路由器（不管有没有拿到 IP 地址），本机会收到很多局域网内的广播包。通过 busyboxifconfig eth0 查看 RX packets 应该不为 0。如果为 0 表示 RX 不通，RX 通路有异常，需要排查。

```
busyboxifconfig eth0
eth0      Link encap:EthernetHWaddr 7E:13:69:61:1C:32
inet6addr: fe80::7c13:69ff:fe61:1c32/64 Scope:Link
```

```
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:83 dropped:0 overruns:0 frame:83
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2188 (2.1 KiB)
          Interrupt:51
```

A) 节调整 rx_delay，看是否有改善

B) 硬件排查原理图是否正确，RX 通路上的 PHY，变压器，RJ45 座子是否正常

C) clock 精度存在问题，PHY 所用的 25M 晶体的频偏要求在 20ppm 以内

4.6 TX 发送异常

如果出现 RX 数据正常，但是 TX 一直发送不出去（表现出的现象是：无法获取到 IP 地址，或者配置静态 IP 地址后无法上网）。

```
buysboxifconfig eth0
// RX 接收数据正常
RX packets:341 errors:0 dropped:0 overruns:0 frame:0
// TX 一直发送不去（TX packets 数目很少）
TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
```

（只要驱动发送成功，TX packets 就会统计，所以 TX packets 看到的是非 0，但是有可能 GMAC 没有成功发送到 PHY，或 PHY 没有成功发送出去）

1. 调整 tx_delay，看是否有改善
2. 调整驱动强度
3. 硬件排查原理图是否正确，TX 通路上的 PHY，变压器，RJ45 座子是否正常
4. clock 精度存在问题，PHY 所用的 25M 晶体的频偏要求在 20ppm 以内

4.7 以太网吞吐率异常

例如 TX 或 RX 吞吐率较低，或不稳定。

A)调整 tx_delay 和 rx_delay，看是否有改善

B)调整驱动强度配置

C)检查硬件 RMII/RGMII 接口走线是否过长

5 PHY 寄存器读写调试

系统中提供以下节点供 phy 寄存器读写：

1. 外部 phy:

   /sys/bus/mdio_bus/devices/stmmac-0:01/phy_reg

   /sys/bus/mdio_bus/devices/stmmac-0:01/phy_regValue

2. 内部 phy:

   /sys/bus/mdio_bus/devices/stmmac-1:01/phy_reg

   /sys/bus/mdio_bus/devices/stmmac-1:01/phy_regValue

3. 举例，如果要读写外部 phy0x01 寄存器，操作如下：

   ```
   echo 1 >/sys/bus/mdio_bus/devices/stmmac-0:01/phy_reg
   #读寄存器 0x01
   cat/sys/bus/mdio_bus/devices/stmmac-0:01/phy_regValue
   #写 0x10 到寄存器 0x01
   echo 0x10 >/sys/bus/mdio_bus/devices/stmmac-0:01/phy_regValue
   ```

