S32G MAC Address Storage

by John Li (nxa08200)

本文说明在S32G GMAC与PFE EMAC的MAC地址是如何获得的，以及如何储存一个MAC地址到fuse中及如何使用。

本文以Linux BSP38来说明。

历史	说明	作者
V1	● 创建本文	John.Li

目录

1 需要的软件

软件/工具	名称	说明
Linux BSP	BSP38	从www.nxp.com/s32g 个人帐号中下载BSP38 Linux 相关文档。

2 背景说明

S32G 作为网关芯片，网络相关应用很重要，而网络应用经常需要用到 MAC 地址，所以需要说明 S32G 的 MAC 地址配置方法，包括 PFE EMAC 与 GMAC。

而作为汽车局域网网关芯片，通常情况下会通过外部 Tbox 或者外部交换机来连接互连网，所以一般配置实验性质的 MAC 地址就能满足局域网应用了。

而如果要使用某个 MAC 来连接互连网，一般来讲，需要向 IEEE 申请购买 MAC 地址段，然后将 MAC 地址烧到板子上的 NVM 上，也就是意味作需要与板子上的镜像分离，多数情况下会采用储存到外部 EEPROM 中，而如果芯片提供 FUSE MAC 地址储存的功能，则可以节省一颗 EEPROM。

本文主要说明 S32G MAC 地址的配置方法，以及如何将一个 MAC 地址储存在 FUSE 中并使用，本文以 GMAC 为例。

3 PFE eMAC MAC 地址说明

3.1 DTS 配置

参考源文件：linux\arch\arm64\dts\freescale\s32g-pfe.dtsi

```
soc {
        pfe: pfe@46000000 {
                compatible = "nxp,s32g-pfe";
…
phy-names = "emac0_xpcs", "emac1_xpcs", "emac2_xpcs";
…
/* Network interface 'pfe0' */
                pfe_netif0: ethernet@10 {
```

```
                compatible = "nxp,s32g-pfe-netif";
…
                local-mac-address = [ 00 04 9F BE EF 00 ];
            nxp,pfeng-if-name = "pfe0";
…
/* Network interface 'pfe1' */
            pfe_netif1: ethernet@11 {
                compatible = "nxp,s32g-pfe-netif";
…
                local-mac-address = [ 00 04 9F BE EF 01 ];
             nxp,pfeng-if-name = "pfe1";
…
/* Network interface 'pfe2' */
            pfe_netif2: ethernet@12 {
                compatible = "nxp,s32g-pfe-netif";
…
                local-mac-address = [ 00 04 9F BE EF 02 ];
           nxp,pfeng-if-name = "pfe1";
…
```

所以 DTS 代码中已经配置了三个 eMAC 的 MAC 地址：local-mac-address。

S32g-pfe-slave.dtsi 中配置与之类似。

3.2 源代码说明

参考源文件：sw\linux-pfeng\pfeng-drv.c

```
pfeng_drv_probe
|-> pfeng_dt_create_config
        /* MAC eth address */
#if LINUX_VERSION_CODE < KERNEL_VERSION(5,13,0)
        if (!IS_ERR_OR_NULL(of_get_mac_address(child))) {
            memcpy(netif_cfg->macaddr, (u8 *)of_get_mac_address(child), ETH_ALEN);
            HM_MSG_DEV_INFO(dev, "DT mac addr: %pM", netif_cfg->macaddr);
        }
#else
        if (!of_get_mac_address(child, netif_cfg->macaddr))
```

```
                HM_MSG_DEV_INFO(dev, "DT mac addr: %pM", netif_cfg->macaddr);
#endif
|   |-> of_get_mac_address
|   |   |-> of_get_mac_addr(np, "local-mac-address", addr);
```

所以 Linux 驱动会从 DTS 中的 local-mac-address 项中读出 MAC 地址来使用。

3.3 测试

将 bsp38/ fsl-image-auto-s32g399ardb3.sdcard 烧写在 TFcard 中，RDB3 设置为 SDcard 启动模式，连接电源，串口，上电，进入 shell 后，输入：

```
dmesg |grep pfeng
[  7.884483] pfeng 46000000.pfe pfe0 (uninitialized): setting MAC addr: 00:01:be:be:ef:11
[   7.886455] pfeng 46000000.pfe pfe1 (uninitialized): setting MAC addr: 00:01:be:be:ef:22
[   7.888209] pfeng 46000000.pfe pfe2 (uninitialized): setting MAC addr: 00:01:be:be:ef:33
ifconfig
pfe0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet6 fe80::201:beff:febe:ef11  prefixlen 64  scopeid 0x20<link>
…
pfe1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      ether 00:01:be:be:ef:22  txqueuelen 1000  (Ethernet)
…
pfe2: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      ether 00:01:be:be:ef:33  txqueuelen 1000  (Ethernet)
```

4 GMAC0 MAC 地址说明

4.1 DTS 配置

参考源文件：linux\arch\arm64\dts\freescale\s32cc.dtsi

```
gmac0: ethernet@4033c000 {
                compatible = "nxp,s32cc-dwmac";
```

所以 DTS 代码中没有配置了 GMAC0 的 MAC 地址。

4.2 源代码说明

参考源文件：linux\drivers\net\ethernet\stmicro\stmmac\dwmac-s32cc.c

s32cc_dwmac_probe

|->stmmac_probe_config_dt

| |->of_get_mac_address //如之前 DTS 中没有配置，所以此外获取不到 MAC 地址

| |->eth_zero_addr

|->stmmac_dvr_probe

| |-> stmmac_check_ether_addr

is_valid_ether_addr//先判断是否已经有 mac 地址，如之前所说，没有

| | |->stmmac_get_umac_addr//会先尝试从 gmac 寄存器中读 MAC 地址

is_valid_ether_addr //如果 gmac 寄存器也没有读到 MAC 地址

| | |-> eth_hw_addr_random

| | | |-> eth_random_addr

所以 Linux 驱动会为 GMAC 生成一个随机的 MAC 地址，如下：

dmesg |grep mac

[5.195113] s32cc-dwmac 4033c000.ethernet: device MAC address 3a:e5:32:73:16:b6

这个值每次启动，都会不一样。而一般来讲，MAC 地址需要确定，甚至指定，随机的 MAC 地址往往不便使用，所以如下 Systemd 服务会再次固定 MAC 地址。

4.3 SystemD 脚本

板子起来后参考脚本：

cat /lib/systemd/network/99-default.link

```
[Match]
OriginalName=*

[Link]
NamePolicy=keep kernel database onboard slot path
AlternativeNamesPolicy=database onboard slot path
MACAddressPolicy=persistent
```

说明文档如下：

https://www.freedesktop.org/software/systemd/man/latest/systemd.link.html

[Link] Section Options

MACAddressPolicy=

The policy by which the MAC address should be set. The available policies are:

persistent

If the hardware has a persistent MAC address, as most hardware should, and if it is used by the kernel, nothing is done. Otherwise, a new MAC address is generated which is guaranteed to be the same on every boot for the given machine and the given device, but which is otherwise random. This feature depends on

ID_NET_NAME_* properties to exist for the link. On hardware where these properties are not set, the generation of a persistent MAC address will fail.

所以当 MACAddressPolicy 设置为 persistent 的话，Systemd 的 udev 服务将无视驱动上传的随机 MAC，而重新配置为一个固定 MAC，对于上传的固定 MAC 地址，则保持。如下：

```
ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 0a:b1:29:bf:72:48  txqueuelen 1000  (Ethernet)
```

而这个默认固定 MAC 值，来自于 SystemD 源代码：systemd\src\udev\net\link-config.c

```
link_generate_new_hw_addr
|->net_get_unique_predictable_data
|    |->net_get_unique_predictable_data_from_name(name, &HASH_KEY, ret);
|    |    |->sd_id128_get_machine
|    |    |    |->id128_read("/etc/machine-id", ID128_FORMAT_PLAIN | ID128_REFUSE_NULL, &saved_machine_id);
/* Fetch some persistent data unique to this machine */
/* Let's hash the machine ID plus the device name. We use
        * a fixed, but originally randomly created hash key here. */
```

所以默认 MAC 地址是来自于是/etc/machine-id + 网口名称的 一个 hash 值。

4.4 固定 GMAC MAC 地址的修改办法

如果需要指定 GMAC0 MAC 地址为所需要的值，有两种方法：

- 从驱动上报固定地址，则需要在 GMAC 的 DTS 中配置 local-mac-address 项。

Linux\arch\arm64\boot\dts\freescale\s32cc.dtsi：

```
gmac0: ethernet@4033c000 {
…
local-mac-address = [ 00 01 02 03 04 05 ];
```

测试如下：

```
ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:01:02:03:04:05  txqueuelen 1000  (Ethernet)
```

- 根据 systemd.link 文档说明：

```
MACAddress=
The interface MAC address to use. For this setting to take effect, MACAddressPolicy= must either be unset, empty, or "none".
```

所以修改/lib/systemd/network/99-default.link：

```
[Match]
OriginalName=eth0
[Link]
NamePolicy=keep kernel database onboard slot path
AlternativeNamesPolicy=database onboard slot path
MACAddress=12:34:56:78:90:ab
```

修改后重启，如下：

```
dmesg |grep mac
[    5.195333] s32cc-dwmac 4033c000.ethernet: device MAC address 06:9e:59:88:d2:f1
ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 12:34:56:78:90:ab  txqueuelen 1000  (Ethernet)
```

5 用 Uboot 命令烧写 FUSE MAC 地址项

参考 S32G 附件 S32G3_fuse_map.xlsx：

0072	0002	0002	MAC_ADDR_LOCK	248	MAC0_ADDR[31:24]	MAC0_ADDR[23:16]	MAC0_ADDR[15:8]	MAC0_ADDR[7:0]
0076	0002	0003	MAC_ADDR_LOCK	24C	**NXP Reserved**	**NXP Reserved**	MAC0_ADDR[47:40]	MAC0_ADDR[39:32]

所以从 fuse bank 2 的 word 2/3 中就可以读出 fuse 中的 mac 地址，注意目前只提供一组 mac 地址，所以此方法只能用于一个 mac。

测试方法：

//先读 UID 有值，证明 fuse read 功能正常

```
=> fuse read 0 4 1
Reading bank 0:

Word 0x00000004: 5563b4d5
=> fuse read 0 5 1
Reading bank 0:

Word 0x00000005: 1c12180b
```

//再读 MAC fuse，默认为 0

```
=> fuse read 2 2 1
Reading bank 2:
```

```
Word 0x00000002: 00000000
=> fuse read 2 3 1
Reading bank 2:

Word 0x00000003: 00000000
```

//此外写入 mac fuse，注意要特别小心，输入 y 确认

```
=> fuse prog 2 2 0x00112244
Programming bank 2 word 0x00000002 to 0x00112244...
Warning: Programming fuses is an irreversible operation!
        This may brick your system.
        Use this command only if you are sure of what you are doing!

Really perform this fuse programming? <y/N>
Y
=> fuse prog 2 3 0x00005566
Programming bank 2 word 0x00000003 to 0x00005566...
Warning: Programming fuses is an irreversible operation!
        This may brick your system.
        Use this command only if you are sure of what you are doing!

Really perform this fuse programming? <y/N>
Y
```

//注意此外需要断电重启，才会写入，然后检查是否写入成功：

```
=> fuse read 2 2 1
Reading bank 2:

Word 0x00000002: 00112244
=> fuse read 2 3 1
Reading bank 2:

Word 0x00000003: 00005566
```

这样就可以将需要量产使用的 MAC 地址烧写到 S32G 的 Fuse 中(需要将此功能实现在量产烧写工具的 Uboot 中，或者采用其它离线烧写的办法)。

6 修改为从 fuse 中获得 GMAC0 MAC 地址

6.1 Uboot 代码修改

本节说明如何修改 Uboot 源代码，使用 fuse 中读出来的 MAC 地址，以 GMAC 驱动为例：

修改文件：uboot/drivers/net/dwc_eth_qos_core.c

```
static const struct eth_ops eqos_ops = {
    …
    .write_hwaddr = eqos_write_hwaddr,
    .read_rom_hwaddr          = eqos_read_rom_hwaddr, //增加一条从 rom 中读取 mac 地址函数。
};
```

然后再增加函数：

```
static int eqos_read_rom_hwaddr(struct udevice *dev)
{
    struct eth_pdata *plat = dev_get_platdata(dev);
    s32cc_get_mac_from_fuse(plat->enetaddr);
    return !is_valid_ethaddr(plat->enetaddr);
}
```

再增加以下调用函数：

```
void s32cc_get_mac_from_fuse(unsigned char *mac)
{
    u32 val[2] = {};
    int ret;
```

//fuse map 说明见前章

```
    ret = fuse_read(2, 2, &val[0]);
    if (ret)
            goto err;
    ret = fuse_read(2, 3, &val[1]);
    if (ret)
            goto err;
    mac[0] = val[0];
    mac[1] = val[0] >> 8;
```

```
    mac[2] = val[0] >> 16;
    mac[3] = val[0] >> 24;
    mac[4] = val[1];
    mac[5] = val[1] >> 8;
    debug("%s: %02x.%02x.%02x.%02x.%02x.%02x\n",
          __func__, mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
    return;
err:
    memset(mac, 0, 6);
    printf("%s: fuse read err: %d\n", __func__, ret);
}
```

6.2 Uboot 写 MAC 寄存器说明

参考源文件：uboot/net/eth-uclass.c

```
.post_probe = eth_post_probe
/* Check if the device has a MAC address in ROM */
|->eth_get_ops(dev)->read_rom_hwaddr(dev);
|->eth_write_hwaddr
|   |->eth_get_ops(dev)->write_hwaddr(dev);= eqos_write_hwaddr
```

所以 Uboot 会将 fuse mac 中读出的 mac 地址写入 GMAC 寄存器中，而通过之前分析，Linux 的 GMAC 驱动会先尝试从 GMAC 寄存器中读取 MAC 地址，再用随机地址，所以实现了修改 Uboot 驱动后，也可以修改掉 Linux 驱动的 MAC 地址。

6.3 测试

Uboot 打印：

```
s32cc_get_mac_from_fuse: 44.22.11.00.66.55
pri
ethaddr=44:22:11:00:66:55
```

内核命令：

```
ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 44:22:11:00:66:55  txqueuelen 1000  (Ethernet)
```