S32G PFE master/slave 配置

本文对S32G PFE Master/Slave不同的使用场景进行不同的配置。

历史	说明	作者
V1	● 创建本文	● Jerry Huang

目录

1 PFE 模块

PFE 是 S32G 系列芯片一个用于网络通信的模块，通过三个 EMAC 接口和外部网络相连，同时有四个 HIF，可以和内部不同 host 进行数据交换，比如 A 核和 M 核可以通过某一个 EMAC 同时和外部建立起网络连接。

针对不同的硬件设计和应用场景，master 和 salve 端可以有不同的配置。

1.1 参考资料

以 S32G3 RDB3 为例：

序号	资料	说明	如何获取
1	S32G3 BSP36 Linux Release. ● S32G3_LinuxBSP_36.0_User_Manual.pdf	根据文档构建 Yocto Linux 编译环境	www.nxp.com/s32g 通过 bundle 下载 Linux 相关文档
2	S32G3 MCAL RTD ● SW32G_RTD_4.4_4.0.1_D2302.exe	RTD MCAL	www.nxp.com/s32g 通过 bundle 下载安装包
3	S32G PFE MCAL Driver ● PFE-DRV_S32G_M7_MCAL_1.1.0.zip	PFE MCAL driver	www.nxp.com/s32g 通过 bundle 下载
4	S32G PFE Firmware ● PFE-FW_S32G_1.6.0.zip	PFE firmware	www.nxp.com/s32g 通过 bundle 下载
5	S32G_PFE_Master_Slave_Simple_Demo_V1_2023_4_21.pdf	应用手册	通过 NXP 社区下载： https://community.nxp.com/pwmxy87654/attachm

1.2 版本匹配说明

参考文档《PFE-DRV_S32G_M7_MCAL_1.1.0_ReleaseNotes.pdf》说明：

Compatible firmware: PFE-FW RTM 1.6.0

Driver compatible with this version of MCAL Master driver (for Master-Slave scenarios):

PFE LINUX RTM 1.3.0

PFE master/slave configurations

Compatible RTD: SW32G_RTD_4.4_4.0.1_D2302

2 编译需要的 Image 文件

PFE master 和 slave 可以运行在 M 核端，也可以运行在 A 核端，或者同时运行在 M 核或 A 核端。文档中以 PFE master 运行在 M 核，slave 运行在 A 核 Linux 中作为例子，如果需要别的使用模式，请参考相关文档。

M 核配置 PFE master，请参考文档 S32G_PFE_Master_Slave_Simple_Demo_V1_2023_4_21.pdf。

编译 PFE master 驱动程序和 PFE slave 驱动程序，可以有两种方式。

2.1 使用 Yocto 编译 PFE 驱动程序

参考文档《S32G3_LinuxBSP_36.0_User_Manual.pdf》说明：

7.3.4 Build by Auto Linux BSP

To get the PFE standalone driver integrated into the NXP Auto Linux BSP, the following two lines must be added to the conf/local.conf file:

```
DISTRO_FEATURES:append = " pfe"
NXP_FIRMWARE_LOCAL_DIR = "/path/to/firmware/binaries/folder"
```

where `/path/to/firmware/binaries/folder` is the local path to the local s32g_pfe_class.fw fw file and optionally s32g_pfe_util.fw.

To build PFE multi instance driver and populate it, the following should be appended to DISTRO_FEATURES:

```
DISTRO_FEATURES:append = " pfe pfe-slave"
```

When `pfe` feature is added without `pfe-slave` feature, the populated driver will be standalone. If both features `pfe` and `pfe-slave` are present, PFE drivers are built as multi instance. Feature `pfe-slave` can be used without `pfe` feature, when the PFE master driver runs in different OS.

NXP_FIRMWARE_LOCAL_DIR：指向 PFE 固件所在目录。

然后使用 Yocto 命令编译 pfe 和 pfe-slave 模块，或者编译整个 S32G3 Image，并烧写到 SD 卡。

2.2 使用 Standalone 编译 PFE 驱动程序

PFE master/slave 驱动程序可以从 github 上获取：https://github.com/nxp-auto-linux/pfeng.git，切换到 tag：BLN_PFE-DRV_S32G_A53_LNX_1.3.0。

设置好编译环境，比如 ARCH，CROSS_COMPILE, path 等（使用 Yocto 生成的 toolchain，CROSS_COMPILE="aarch64-fsl-linux-"），然后进入目录 pfeng/sw/linux-pfeng。

执行命令编译 PFE master 驱动：

make PFE_CFG_PFE_MASTER=1 KERNELDIR = "path/to/your/Linux"

执行命令编译 PFE slave 驱动：

make PFE_CFG_PFE_MASTER=1 KERNELDIR = "path/to/your/Linux"

其中，KERNELDIR 指向 Linux 源码目录。

得到 PFE master 和 slave 端的驱动 pfeng.ko，pfeng-slave.ko。

1. 将需要的 PFE 固件 s32g_pfe_class.fw，s32g_pfe_util.fw 拷贝到目标板上 SD 卡文件系统目录/lib/firmware/.
2. 将编译好的 PFE master/slave 驱动成 pfeng.ko，pfeng-slave.ko 拷贝到目标板上 SD 卡文件系统中。

2.3 使用 Standalone 编译 PFE DTS 文件

使用默认的 DTS 文件 s32g399a-rdb3.dts，PFE 所有 EMAC 都会使能为 master 模式。如果 A 核运行 PFE slave，或者 A 核同时运行 master 和 slave 模式，需要对 DTS 文件进行修改，然后编译成二进制文件。

从地址https://github.com/nxp-auto-linux/linux.git 下载 Linux 源码，并切换到 branch：release/bsp36.0-5.10.158-rt。

设置好编译环境，修改需要的 DTS 文件，比如 s32g-pfe-slave.dtsi 或者 s32g399a-rdb3-pfems.dts，然后使用下面的命令进行编译（以 s32g399a-rdb3-pfems.dts 为例）：

make freescale/s32g399a-rdb3-pfems.dtb

然后将编译好的 s32g399a-rdb3-pfems.dtb 拷贝到目标板上 SD 卡的第一个分区，并改名为 s32g399a-rdb3.dtb，重新启动目标板。

3 PFE Master/Slave 配置

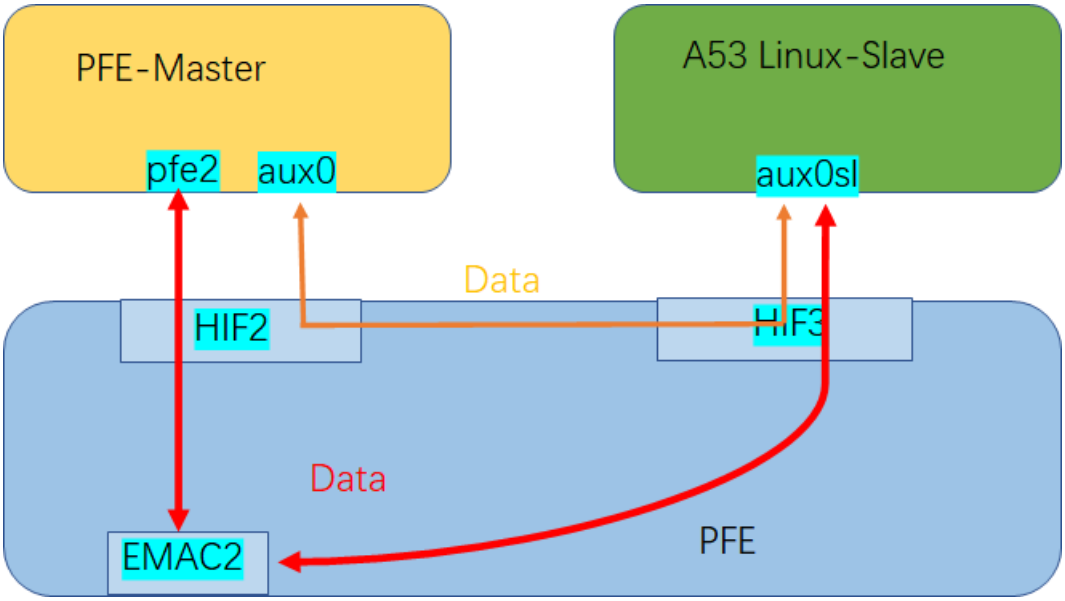
PFE master 和 slave 可以在 M 核配置，也可以在 A 核配置，或则同时在 M 核或者 A 核配置。一个典型的应用是 M 核配置为 PFE master，A 核配置为 PFE slave。

在本文档中，以 S32G399ARDB3 作为验证开发板，使用 EMAC2 作为例子。PFE slave 在 A 核端进行配置，使用 HIF3 接口；PFE master 在 M 核端配置，pfe2 和 aux0 共享 HIF2 接口。

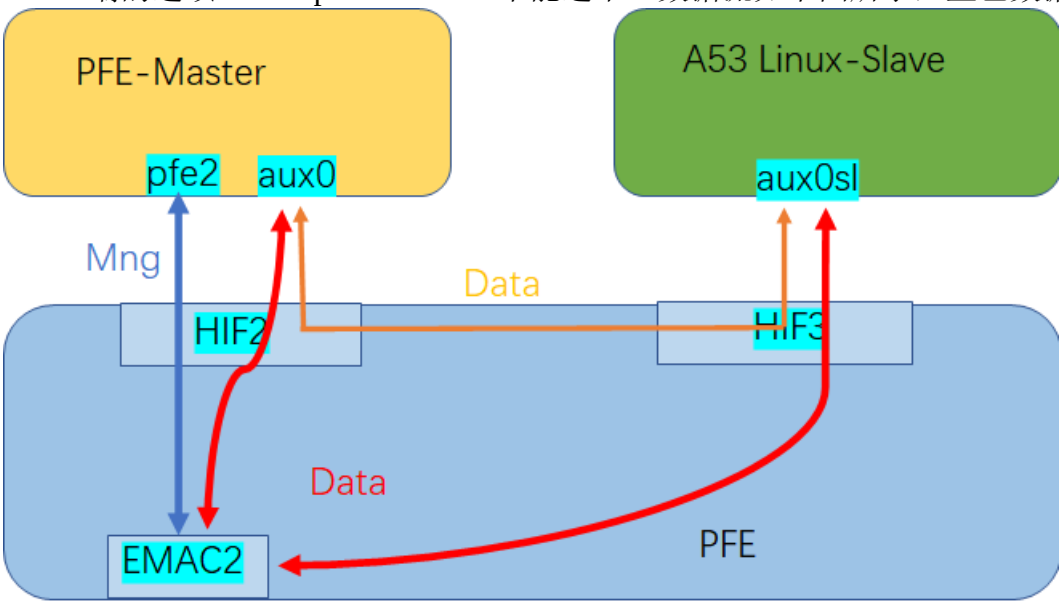
3.1 在 A 核端配置 PFE slave - aux0sl 接口

如下图所示，PFE master运行在M核端，其中pfe2接口通过EMAC2与外部网络进行通信（红色数据流），aux0只与PFE slave端的aux0sl进行内部通信（橙色数据流）。此时Master端的选项“Accept all traffic”需要选中。

PFE slave只有一个接口aux0sl，可以通过AUX控制器与master端的aux0进行通信（橙色数据流），也可以通过EMAC2与外部网络进行通信（红色数据流）。



如果PFE master端的接口pfe2只收发管理报文，如PTP报文，而其余所有的报文都由aux0处理，那么master端的选项“Accept all traffic”不能选中。数据流如下图所示，蓝色数据流为管理报文。



PFE master/slave configurations

下面为Linux DTS中配置PFE slave（默认情况下，pfesl0，pfesl1和pfesl2都是使能的）：

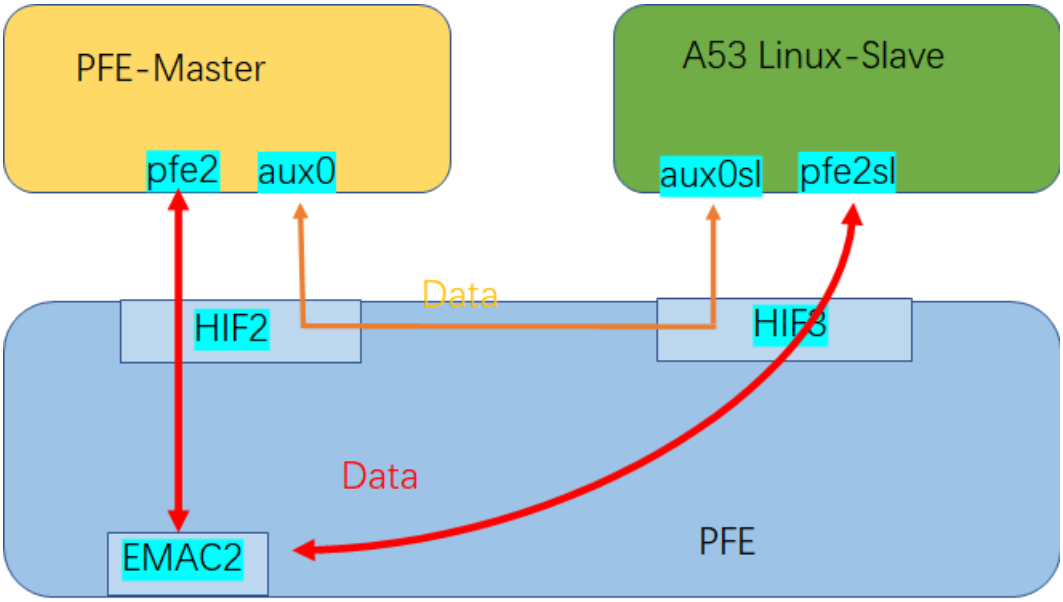
```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -14,6 +14,18 @@
        status = "okay";
 };

+&pfesl_netif0 {
+       status = "disabled";
+};
+
+&pfesl_netif1 {
+       status = "disabled";
+};
+
+&pfesl_netif2 {
+       status = "disabled";
+};
+
 &pfesl_aux0 {
        status = "okay";
 };
```

3.2 在 A 核端配置 PFE slave - pfe2sl 和 aux0sl 接口

结合PFE master的不同配置，可以形成不同的组合。

1. 如下图所以，aux0与aux0sl通过AUX控制器在S32G3内部通信（橙色数据流）；pfe2和pfe2sl通过EMAC2与外部网络进行通信（红色数据流）。Master 端的选项“Accept all traffic”需要选中。



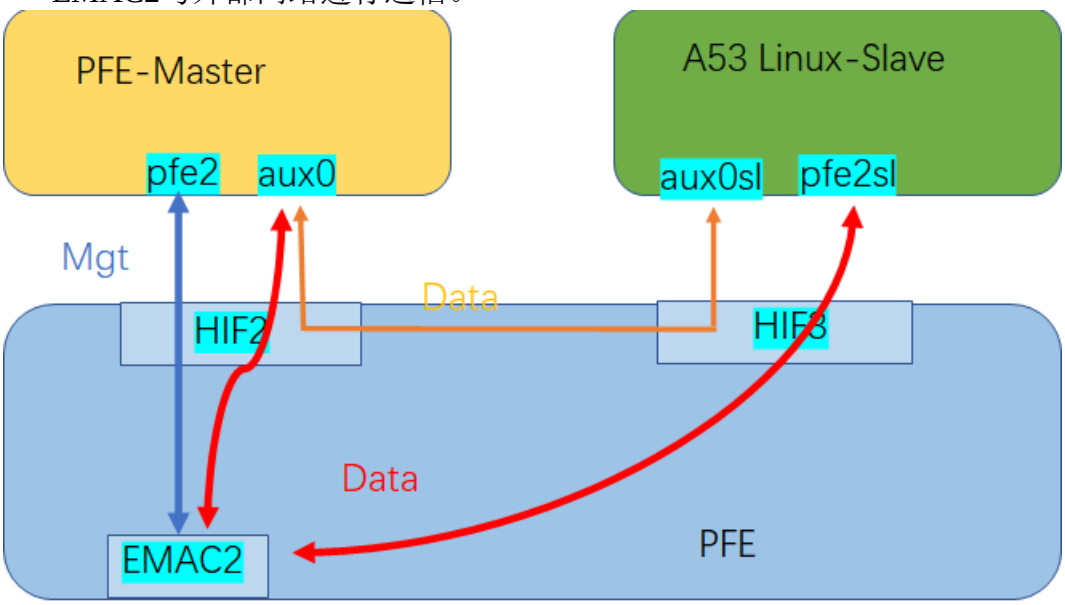
下面为Linux DTS中配置PFE slave（默认情况下，pfesl0，pfesl1和pfesl2都是使能的）：

PFE master/slave configurations

```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -14,6 +14,14 @@
        status = "okay";
 };

+&pfesl_netif0 {
+       status = "disabled";
+};
+
+&pfesl_netif1 {
+       status = "disabled";
+};
+
 &pfesl_aux0 {
        status = "okay";
 };
```

2. PFE master的pfe2设置为只处理管理报文（如蓝色数据流），Master端的选项“Accept all traffic”不能选中；Master端的aux0既通过AUX控制器和slave端的aux0sl进行通信，同时也通过EMAC2与外部网络进行通信。而slave端的aux0sl只进行内部通信，pfe2sl通过EMAC2与外部网络进行通信。



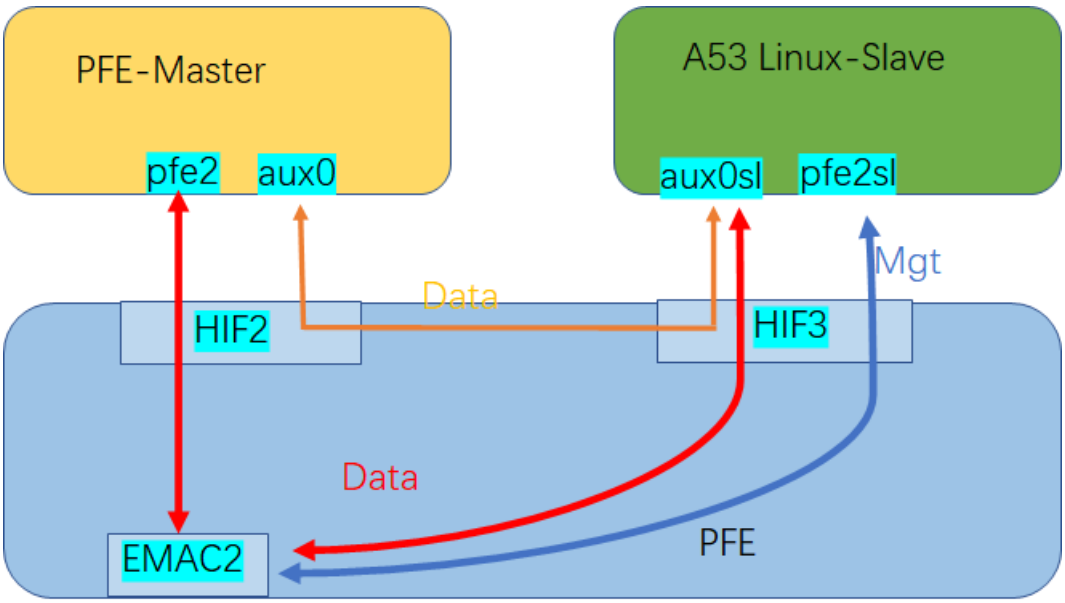
下面为Linux DTS中配置PFE slave（默认情况下，pfesl0，pfesl1和pfesl2都是使能的）：

PFE master/slave configurations

```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -14,6 +14,14 @@
        status = "okay";
 };

+&pfesl_netif0 {
+       status = "disabled";
+};
+
+&pfesl_netif1 {
+       status = "disabled";
+};
+
 &pfesl_aux0 {
        status = "okay";
 };
```

3. PFE slave的pfe2sl设置为只处理管理报文（如下图蓝色数据流），Master端的选项“Accept all traffic”需要选中；在PFE slave端pfe2sl的节点中增加一个属性：nxp,pfeng-netif-mode-mgmt-only。此时master端的pfe2通过EMAC2与外部网络通信，而aux0通过AUX控制器与slave端的aux0sl进行通信。而slave端的pfe2sl只能处理管理报文，如PTP报文；而aux0sl既可以通过AUX进行内部通信，也可以通过EMAC2与外部网络进行通信。



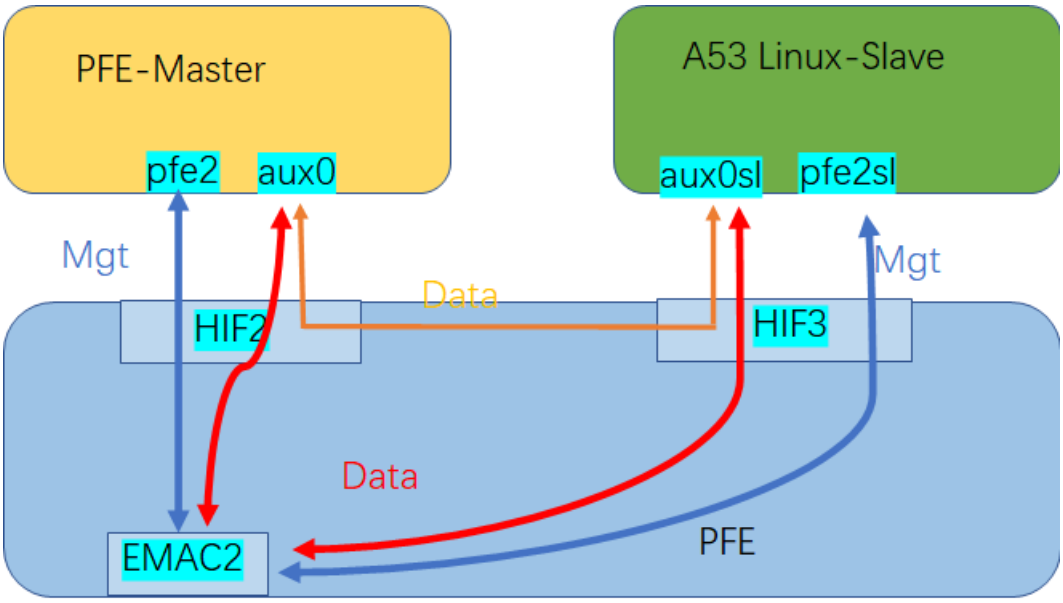
下面为Linux DTS中配置PFE slave（默认情况下，pfesl0，pfesl1和pfesl2都是使能的）：

PFE master/slave configurations

```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -14,6 +14,18 @@
        status = "okay";
 };

+&pfesl_netif0 {
+       status = "disabled";
+};
+
+&pfesl_netif1 {
+       status = "disabled";
+};
+
+&pfesl_netif2 {
+       nxp,pfeng-netif-mode-mgmt-only;
+};
+
 &pfesl_aux0 {
        status = "okay";
 };
```

4. 将master端的pfe2和slave端的pfe2sl都设置为处理管理报文（如下图蓝色数据流所示，Master端的选项“Accept all traffic”不选中；同时在PFE slave 端pfe2sl的节点中增加一个属性：nxp,pfeng-netif-mode-mgmt-only。此时master端的aux0和slave端的aux0sl可以同时进行内部通信，也可以通过EMAC2与外部网络进行通信。



下面为Linux DTS中配置PFE slave（默认情况下，pfesl0，pfesl1和pfesl2都是使能的）：

PFE master/slave configurations

```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -14,6 +14,18 @@
        status = "okay";
 };

+&pfesl_netif0 {
+       status = "disabled";
+};
+
+&pfesl_netif1 {
+       status = "disabled";
+};
+
+&pfesl_netif2 {
+       nxp,pfeng-netif-mode-mgmt-only;
+};
+
 &pfesl_aux0 {
        status = "okay";
 };
```

3.3 PFE master 运行在 A 核端

之前两个章节中涉及的 PFE master，都是运行在 M 核端。如果 PFE master 和 PFE salve 同时运行在 A 核端，PFE slave 端的配置和前面两节的配置一致，PFE Master 配置需要对 DTS 进行修改。

下面为Linux DTS中配置PFE master（默认情况下，pfe0，pfe1和pfe2都是使能的）：

```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -10,6 +10,14 @@
 #include "s32g399a-rdb3.dtsi"
 #include "s32g-pfe-slave.dtsi"

+&pfe_netif0 {
+       status = "disabled";
+};
+
+&pfe_netif1 {
+       status = "disabled";
+};
+
 &pfe_aux0 {
        status = "okay";
 };
```

或者 pfe2 仅仅处理管理报文：

```
--- a/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
+++ b/arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dts
@@ -10,6 +10,18 @@
 #include "s32g399a-rdb3.dtsi"
 #include "s32g-pfe-slave.dtsi"

+&pfe_netif0 {
+       status = "disabled";
+};
+
+&pfe_netif1 {
+       status = "disabled";
+};
+
+&pfe_netif2 {
+       nxp,pfeng-netif-mode-mgmt-only;
+};
+
 &pfe_aux0 {
        status = "okay";
 };
```

4 PFE master/slave VLAN bridge 配置

PFE 的一个典型的使用模式是 VLAN bridge，以第三章的配置为例，将 EMAC2，HIF2 和 HIF3 组成一个 VLAN bridge。

```
libfci_cli bd-update --vlan=1 --uh=0 --um=1 --mh=0 --mm=1
libfci_cli bd-insif --vlan=1 --i=emac2 --tag=ON
libfci_cli bd-insif --vlan=1 --i=hif3 --tag=ON
libfci_cli bd-insif --vlan=1 --i=hif2 --tag=ON
libfci_cli phyif-update --i=emac2 --enable --promisc=ON --mode=VLAN_BRIDGE
libfci_cli phyif-update --i=hif3 --enable --promisc=ON --mode=VLAN_BRIDGE
libfci_cli phyif-update --i=hif2 --enable --promisc=ON --mode=VLAN_BRIDGE
```

5 PFE master/multi-slave 模式

PFE 支持多个 slave 的模式，比如在 M7_0 核上运行 PFE master，M7_1 核上运行 PFE slave，同时在 A 核端也运行 PFE slave。在 M 核上配置 PFE master 或者 slave 请参照相关文档。