S32G Compiled PFE Driver into Kernel

by John Li (nxa08200)

This application doc explains how to modify the PFE driver from the original driver module mode to the compiled into kernel mode on the S32G3 RDB3 board, in order to accelerate the loading speed of the PFE driver and meet the requirements of Linux fast startup. It will support:

- Compile the Slave driver to the kernel.
- Compile the Master driver to the kernel.

The software version is BSP37 (PFE Linux Driver 1.4.0, FW version 1.7.0).

History	Comment	Author
V1	● Create the doc	John.Li
V1	● Translate to English	John.Li

Contents

1 Related Materials

Materials	Name	Comments
Linux BSP	BSP37(include PFE driver version 1.4.0) Refer doc: <<S32G2_LinuxBSP_37.0_User_Manual.pdf>>	Downlad from www.nxp.com/s32g
PFE Linux driver source	Refer doc: <<PFE_S32G_A53_LNX_UserManual.pdf>>	git clone https://github.com/nxp-auto-linux/pfeng/
App notes	S32G_GitServer_Migrate_V*.pdf Include PFE Standalone compiling method	https://community.nxp.com/t5/NXP-Designs-Knowledge-Base/S32G-GitServer-Migrate-chinese-version/ta-p/1689830

2 Current PFE Driver

Refer to the document PFE Linux Driver 1.4.0 <<PFE_S32G_A53_LNX_UserManual. Pdf>> and the document <<S32G_GitServer-Migrate_V *. Pdf>> to learn about the download and standalone compilation of PFE. Please note that document 2 explains how to modify to support different compilers. The operation is as follows:

git clone https://github.com/nxp-auto-linux/pfeng/

cd pfeng/

git checkout BLN_PFE-DRV_S32G_A53_LNX_1.4.0

cd sw/linux-pfeng

vi ../build_env.mak

export PFE_CFG_TARGET_ARCH_DEF=PFE_CFG_TARGET_ARCH_aarch64 # See the document for the reason of modification

Compile Slave driver using SDK compiler:

make PFE_CFG_MULTI_INSTANCE_SUPPORT=1 PFE_CFG_PFE_MASTER=0 PFE_CFG_TARGET_ARCH_aarch64=1 KERNELDIR=~/BSP37/standalone/linux/ PLATFORM=~/BSP37/fsl-image-auto-bsp/sdk/sysroots/x86_64-fslbsp-linux/usr/bin/aarch64-fsl-linux/aarch64-fsl-linux all

Compile Master driver:

make PFE_CFG_PFE_MASTER=1 PFE_CFG_TARGET_ARCH_aarch64=1 KERNELDIR=~/BSP37/standalone/linux/ PLATFORM=~/BSP37/fsl-image-auto-bsp/sdk/sysroots/x86_64-fslbsp-linux/usr/bin/aarch64-fsl-linux/aarch64-fsl-linux all

Save the compiled log for future reference.

The default PFE driver is compiled in the driver module mode, and the YOCTO rootfs automatic loading directory located in rootfs is as follows:
/lib/modules/5.10.120-rt70+g0b76731696c1/kernel/drivers/net/ethernet/nxp/pfe/

3 Compiled PFE Slave into Kernel

According to the document mentioned in the previous chapter, download the PFE driver.

3.1 Put the PFE driver source into kernel source

1. Modified: linux\drivers\net\ethernet\Makefile

   obj-$(CONFIG_NET_VENDOR_PFE) += pfe/

   Modified: \drivers\net\ethernet\Kconfig

   source "drivers/net/ethernet/pfe/Kconfig"

   Then create an empty folder under linux\drivers\net\ethernet\ named: pfe。

2. In order to comply with Linux driver source code orgnizaion method, copy all source files and header files of all levels of directories under he pfe source code to linux\drivers\net\ethernet\pfe, forming a single level source files directory:

ls drivers/net/ethernet/pfe/

blalloc.c fci_hm.c hw oal_time.h pfe_class_csr.h pfe_fw_fail_stop_csr.c pfe_host_fail_stop_csr.h pfeng-dt.c pfe_platform_rpc.h

blalloc.h fci_interfaces.c Kconfig oal_time_linux.c pfe_class.h pfe_fw_fail_stop_csr.h pfe_host_fail_stop.h pfeng-ethtool.c pfe_platform_slave.c

bpool.c fci_internal.h libfci.h oal_types.h pfe_compiler.h pfe_fw_fail_stop.h pfe_hw_feature.c pfeng-fw.c pfe_rtable.c

bpool.h fci_l2br.c libfci_linux.c oal_types_linux.h pfe_ct_comp.h pfe_fw_feature.c pfe_hw_feature.h pfeng.h pfe_rtable.h

build_env.mak fci_l2br_domains.c linked_list.h oal_util.h pfe_ct.h pfe_fw_feature.h pfe_idex.c pfeng-hif.c pfe_spd_acc.c

common.c fci_mirror.c Makefile oal_util_linux.c pfe_ecc_err.c pfe_global_wsp.h pfe_idex.h pfeng-hwts.c pfe_spd_acc.h

ct_assert.h fci_mirror.h Makefile.bak.ok oal_util_net.h pfe_ecc_err_csr.c pfe_gpi.c pfe_if_db.c pfeng-mdio.c pfe_spd.c

dummy_main.c fci_msg.h Makefile.org oal_util_net_linux.c pfe_ecc_err_csr.h pfe_gpi_csr.c pfe_if_db.h pfeng-netif.c pfe_spd.h

dummy_mod.mak fci_msg_linux.h oal.h oal_util_net_linux.h pfe_ecc_err.h pfe_gpi_csr.h pfe_l2br.c pfeng-phylink.c pfe_tmu.c

elf.c fci_owner.c oal_irq.h pfe_bmu.c pfe_emac.c pfe_gpi.h pfe_l2br.h pfeng-ptp.c pfe_tmu_csr.c

elf_cfg.h fci_ownership_mask.h oal_irq_linux.c pfe_bmu_csr.c pfe_emac_csr.c pfe_hif.c pfe_l2br_table.c pfeng-slave-drv.c pfe_tmu_csr.h

elf.h fci_qos.c oal_job.h pfe_bmu_csr.h pfe_emac_csr.h pfe_hif_chnl_linux.c pfe_l2br_table_csr.h pfe_parity.c pfe_tmu.h

fci.c fci_routes.c oal_job_linux.c pfe_bmu.h pfe_emac.h pfe_hif_chnl_linux.h pfe_l2br_table.h pfe_parity_csr.c pfe_util.c

fci_connections.c fci_rt_db.c oal_master_if.h pfe_bus_err.c pfe_emac_slave.c pfe_hif_csr.c pfe_log_if.c pfe_parity_csr.h pfe_util_csr.c

fci_core.h fci_rt_db.h oal_mbox.h pfe_bus_err_csr.c pfe_fail_stop.c pfe_hif_csr.h pfe_log_if.h pfe_parity.h pfe_util_csr.h

fci_core_linux.c fci_spd.c oal_mbox_linux.c pfe_bus_err_csr.h pfe_fail_stop_csr.c pfe_hif_drv.h pfe_log_if_slave.c pfe_pe.c pfe_util.h

fci_fp.c fci_spd.h oal_mm.h pfe_bus_err.h pfe_fail_stop_csr.h pfe_hif.h pfe_mac_db.c pfe_pe.h pfe_wdt.c

fci_fp_db.c fifo.c oal_mm_linux.c pfe_cbus.h pfe_fail_stop.h pfe_hif_ring_linux.c pfe_mac_db.h pfe_phy_if.c pfe_wdt_csr.c

fci_fp_db.h fifo.h oal_mutex_linux.h pfe_cfg.h pfe_feature_mgr.c pfe_hif_ring_linux.h pfe_mirror.c pfe_phy_if.h pfe_wdt_csr.h

fci_fp.h fpp_ext.h oal_spinlock_linux.h pfe_cfg.h.bak pfe_feature_mgr.h pfe_hm.c pfe_mirror.h pfe_phy_if_slave.c pfe_wdt.h

fci_fw_features.c fpp.h oal_sync.h pfe_cfg.h.ok2 pfe_fp.c pfe_hm.h pfeng-bman.c pfe_platform_cfg.h

fci_fw_features.h hal.c oal_thread.h pfe_class.c pfe_fp.h pfe_host_fail_stop.c pfeng-debugfs.c pfe_platform.h

fci.h hal.h oal_thread_linux.c pfe_class_csr.c pfe_fw_fail_stop.c pfe_host_fail_stop_csr.c pfeng-drv.c pfe_platform_master.c

3. Create a Kconfig filer under \linux\drivers\net\ethernet\pfe, edit it as follows

```
# Configure the compilation macro definition according to the subdirectory requirements of the previous PFE
#compilation, with default=y
config NET_VENDOR_PFE
        bool "PFE devices"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

config NET_PFE_OAL
        bool "PFE devices oal module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core
```

```
config NET_PFE_BPOOL
        bool "PFE devices bpool module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

config NET_PFE_ELF
        bool "PFE devices elf module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

config NET_PFE_FIFO
        bool "PFE devices fifo module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

config NET_PFE_PFE_PLATFORM
        bool "PFE devices pfe_platform module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

config NET_PFE_COMMON
        bool "PFE devices common module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

config NET_PFE_FCI
        bool "PFE devices fci module"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core


config NET_PFE_LINUX_PFENG
        bool "PFE devices main pfeng driver"
        default y
        help
          If you have a network (Ethernet) card based on PFE IP
          Core
#when compiled the Slave driver, default: MASTER=0
config PFE_CFG_PFE_MASTER
```

```
        int "PFE master"
        default 0
        help
          If you have a network (Ethernet) card based on PFE IP
          Core
# when compiled the Slave driver, default: MULTI_INSTANC=1

config  PFE_CFG_MULTI_INSTANCE_SUPPORT
        int "PFE multi instance"
        default 1
        help
          If you have a network (Ethernet) card based on PFE IP
          Core
```

4. Create the Makefile under linux\drivers\net\ethernet\pfe, edit it as follows:

3.2 Develop the Makefile

1. Configure the compilation macro according to the print of PFE standalone compilation before, for example, when compiling as SLAVE driver, the compiling log printed:

```
… GLOBAL_CCFLAGS="-DPFE_CFG_VERBOSITY_LEVEL=4 -DPFE_CFG_PFE_MASTER
-DPFE_CFG_MASTER_IF=6 -DPFE_CFG_LOCAL_IF=6 -DPFE_CFG_PFE0_IF=6 -DPFE_CFG_PFE1_IF=7
-DPFE_CFG_PFE2_IF=8 -DPFE_CFG_PFE0_PROMISC=1 -DPFE_CFG_PFE1_PROMISC=1
-DPFE_CFG_PFE2_PROMISC=1 -DPFE_CFG_HIF_USE_BD_TRIGGER -DPFE_CFG_BUFFERS_COHERENT
-DPFE_CFG_RTABLE_ENABLE -DPFE_CFG_FCI_ENABLE -DPFE_CFG_GLOB_ERR_POLL_WORKER
-DPFE_CFG_FLEX_PARSER_AND_FILTER -DPFE_CFG_SC_HIF -DPFE_CFG_HIF_TX_FIFO_FIX
-DPFE_CFG_BMU_IRQ_ENABLED=TRUE -DPFE_CFG_SYS_MEM='\""pfe-bmu2-pool"\"'
-DPFE_CFG_BD_MEM='\""pfe-bdr-pool"\"' -DPFE_CFG_RX_MEM='\""pfe_ddr"\"'
-DPFE_CFG_TX_MEM='\""pfe_ddr"\"' -DPFE_CFG_RT_MEM='\""pfe-rt-pool"\"'
-DPFE_CFG_RT_HASH_SIZE=256 -DPFE_CFG_RT_COLLISION_SIZE=256
-DPFE_CFG_CONN_STATS_SIZE=20 -DPFE_CFG_HM_STRINGS_ENABLED -DPFE_CFG_HIF_PRIO_CTRL
-DPFE_CFG_SAFE_IRQ -DPFE_CFG_BMU2_BUF_COUNT=256 -DPFE_CFG_HIF_RING_LENGTH=256
-DPFE_HIF_RX_RING_CFG_LENGTH=256 -DM4_FILE_VERSION_CHECK_HDR=
-DM4_FILE_VERSION_CHECK_SRC="
```

So we configure the compilation macro as:

Notes:

-DPFE_CFG_TARGET_OS_LINUX=TRUE -DPFE_CFG_TARGET_ARCH_aarch64=TRUE, these 2 items need to be added

```
#MASTER and SLAVE driver common parts
EXTRA_CFLAGS = -DPFE_CFG_TARGET_OS_LINUX=TRUE -DPFE_CFG_TARGET_ARCH_aarch64=TRUE
-DPFE_CFG_VERBOSITY_LEVEL=4 -DPFE_CFG_MASTER_IF=6 -DPFE_CFG_LOCAL_IF=6
-DPFE_CFG_PFE0_IF=6 -DPFE_CFG_PFE1_IF=7 -DPFE_CFG_PFE2_IF=8 -DPFE_CFG_PFE0_PROMISC=1
-DPFE_CFG_PFE1_PROMISC=1 -DPFE_CFG_PFE2_PROMISC=1 -DPFE_CFG_MASTER_IF=6
-DPFE_CFG_LOCAL_IF=6 -DPFE_CFG_PFE0_IF=6 -DPFE_CFG_PFE1_IF=7 -DPFE_CFG_PFE2_IF=8
-DPFE_CFG_PFE0_PROMISC=1 -DPFE_CFG_PFE1_PROMISC=1 -DPFE_CFG_PFE2_PROMISC=1
-DPFE_CFG_HIF_USE_BD_TRIGGER -DPFE_CFG_BUFFERS_COHERENT -DPFE_CFG_RTABLE_ENABLE
-DPFE_CFG_FCI_ENABLE -DPFE_CFG_GLOB_ERR_POLL_WORKER
-DPFE_CFG_FLEX_PARSER_AND_FILTER -DPFE_CFG_SC_HIF -DPFE_CFG_HIF_TX_FIFO_FIX
```

```
-DPFE_CFG_BMU_IRQ_ENABLED=TRUE -DPFE_CFG_SYS_MEM='"pfe-bmu2-pool"'
-DPFE_CFG_BD_MEM='"pfe-bdr-pool"' -DPFE_CFG_RX_MEM='"pfe_ddr"' -DPFE_CFG_TX_MEM='"pfe_ddr"'
-DPFE_CFG_RT_MEM='"pfe-rt-pool"' -DPFE_CFG_RT_HASH_SIZE=256
-DPFE_CFG_RT_COLLISION_SIZE=256 -DPFE_CFG_CONN_STATS_SIZE=20
-DPFE_CFG_HM_STRINGS_ENABLED -DPFE_CFG_HIF_PRIO_CTRL -DPFE_CFG_SAFE_IRQ
-DPFE_CFG_BMU2_BUF_COUNT=256 -DPFE_CFG_HIF_RING_LENGTH=256
-DPFE_HIF_RX_RING_CFG_LENGTH=256
#The following is the special part of SLAVE drive
ifeq ($(CONFIG_PFE_CFG_PFE_MASTER),0)
EXTRA_CFLAGS += -DPFE_CFG_PFE_SLAVE -DPFE_CFG_MULTI_INSTANCE_SUPPORT
-DPFE_CFG_SLAVE_HIF_MASTER_UP_TMOUT=1000 -DPFE_CFG_IP_READY_MS_TMOUT=5000
-DPFE_CFG_ERR051211_WORKAROUND_ENABLE
else
# The following is the special part of MASTER drive
EXTRA_CFLAGS += -DPFE_CFG_PFE_MASTER
endif
# This macro is added to remove the date time warning as an error. Compile the warning as follows
{
warning: macro "__DATE__" might prevent reproducible builds [-Wdate-time]
  688 | ct_assert(sizeof(pfe_date_str_t) > sizeof(__DATE__));
}
EXTRA_CFLAGS += -Wno-error=date-time
```

2. According to the original \pfeng-1.4.0\sw\linux-pfeng\Makefile description:

```
pfeng-objs-libs := ../pfe_platform/pfe_platform.o ../oal/oal.o ../elf/elf.o ../fifo/fifo.o ../bpool/bpool.o ../common/co
mmon.o
pfeng-objs-core := pfeng-debugfs.o pfeng-hif.o pfeng-bman.o pfeng-netif.o pfeng-ethtool.o pfeng-hwts.o pfeng-dt.o
pfeng-mdio.o
ifneq ($(PFE_CFG_PFE_MASTER),0)
pfeng-objs := $(pfeng-objs-core) $(pfeng-objs-libs) pfeng-drv.o pfeng-fw.o pfeng-phylink.o pfeng-ptp.o
obj-m += pfeng.o
else
pfeng-slave-objs := $(pfeng-objs-core) $(pfeng-objs-libs) pfeng-slave-drv.o
obj-m += pfeng-slave.o
endif
```

Redesign compilation dependencies:

```
#Note the link order
obj-y :=linux_pfeng.o oal.o bpool.o elf.o fifo.o pfe_platform.o common.o fci.o
# Compile the following *. o files according to the configuration of Kconfig as y
obj-$(CONFIG_PFE_OAL) += oal.o
obj-$(CONFIG_PFE_BPOOL) += bpool.o
obj-$(CONFIG_PFE_ELF) += elf.o
obj-$(CONFIG_PFE_FIFO) += fifo.o
obj-$(CONFIG_PFE_PFE_PLATFORM) += pfe_platform.o
obj-$(CONFIG_PFE_COMMON) += common.o
obj-$(CONFIG_PFE_FCI) += fci.o
obj-$(CONFIG_PFE_LINUX_PFENG) += linux_pfeng.o
```

3. Design the compilation dependency of the original subdirectory files according to the Makefile files of each subdirectory under the original \pfeng-1.4.0\sw :

   Take \pfeng-1.4.0\sw\oal\Makefile as sample: The original compilation source file list is:

```
SRCS += src/oal_mbox_linux.c
SRCS += src/oal_irq_linux.c
SRCS += src/oal_mm_linux.c
SRCS += src/oal_thread_linux.c
SRCS += src/oal_time_linux.c
SRCS += src/oal_util_linux.c
SRCS += src/oal_util_net_linux.c
SRCS += src/oal_job_linux.c
```

Therefore, it is modified as follows:

```
oal-y := oal_mbox_linux.o oal_irq_linux.o oal_mm_linux.o oal_thread_linux.o oal_time_linux.o oal_util_linux.o
oal_util_net_linux.o oal_job_linux.o
```

Follow the above method to add the compilation dependency of other original subdirectories. Note that there are some differences between Master/Slave drivers, such as:

```
ifeq ($(CONFIG_PFE_CFG_PFE_MASTER),0)
        pfe_platform-y :=                pfe_hif_chnl_linux.o \
    ..
else
        pfe_platform-y  :=          pfe_bmu.o \
    ..
ifeq ($(CONFIG_PFE_CFG_MULTI_INSTANCE_SUPPORT),1)
 pfe_platform-y  += pfe_idex.o
endif
endif
```

This completes the Makefile design. The design principle is to try not to modify the source code. For details, please refer to the attached source code package.

3.3 Compilation and Testing

1. Compilation:

```
make s32cc_defconfig
vi .config
CONFIG_NET_VENDOR_PFE=y
CONFIG_NET_PFE_OAL=y
CONFIG_NET_PFE_BPOOL=y
CONFIG_NET_PFE_ELF=y
CONFIG_NET_PFE_FIFO=y
CONFIG_NET_PFE_PFE_PLATFORM=y
CONFIG_NET_PFE_COMMON=y
CONFIG_NET_PFE_FCI=y
CONFIG_NET_PFE_LINUX_PFENG=y
CONFIG_PFE_CFG_PFE_MASTER=0
```

```
CONFIG_PFE_CFG_MULTI_INSTANCE_SUPPORT=1
make -j8
DTC     arch/arm64/boot/dts/freescale/s32g399a-rdb3.dtb
 DTC     arch/arm64/boot/dts/freescale/s32g399a-rdb3-pfems.dtb
OBJCOPY arch/arm64/boot/Image
```

2. Burn the original Linux *.sdcard image

```
sudo dd if= fsl-image-auto-s32g399ardb3.sdcard of=/dev/sd<partition> bs=1M conv=fsync
```

3. On Linux PC, rename the original pfeng.ko auto loading driver module, so that the pfeng.ko module will not be automatically loaded when Linux starts.

```
vmuser@ubuntu:/media/vmuser/fsl-image-auto-s$ cd lib/modules/5.15.96-rt61+gf2b25660adcf/kernel/drivers/net/ethernet/nxp/pfe/
vmuser@ubuntu:/media/vmuser/fsl-image-auto-s/lib/modules/5.15.96-rt61+gf2b25660adcf/kernel/drivers/net/ethernet/nxp/pfe$ ls
pfeng.ko
vmuser@ubuntu:/media/vmuser/fsl-image-auto-s/lib/modules/5.15.96-rt61+gf2b25660adcf/kernel/drivers/net/ethernet/nxp/pfe$ sudo mv pfeng.ko pfeng.ko.bak
vmuser@ubuntu:/media/vmuser/fsl-image-auto-s/lib/modules/5.15.96-rt61+gf2b25660adcf/kernel/drivers/net/ethernet/nxp/pfe$ ls
pfeng.ko.bak
```

You can start it to verify:

```
root@s32g399ardb3:~# dmesg |grep pfe
[    0.000000] OF: reserved mem: initialized node pfebufs@83200000, compatible id shared-dma-pool
root@s32g399ardb3:~# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
   …
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

Have No PFE network device.

4. Replace the image

   Use the Linux PC to bunr Image into Sdcard to replace the original Image, and also copy the dtb into Sdcard.

```
cd /media/boot_s32g3/
cp /mnt/hgfs/share_folder/bsp37/compile_pfe_to_kernel/Image .
cp /mnt/hgfs/share_folder/bsp37/compile_pfe_to_kernel/s32g399a-rdb3-pfems.dtb .
```

5. Hardware configuration: Set the S32G3 RDB3 board to enable the Sdcard boot normally, connect the power supply, serial port, and open the serial port terminal, boot the board.
6. Modify Uboot parameters

After startup, replace the DTB file with one that supports PFE Slave:

```
setenv fdt_file “s32g399a-rdb3-pfems.dtb”
env save
```

Then disable PFE in Uboot:

```
setenv ethact “eth_eqos”
pfeng_mode=none,none,none,none
env save
reset
```

7. Boot Log:

```
root@s32g399ardb3:~# dmesg |grep pfe
[    0.000000] OF: reserved mem: initialized node pfebufs@83200000, compatible id shared-dma-pool
[    0.838262] pfeng-slave 46000000.pfe  slave: PFEng ethernet driver loading ...
[    0.845527] pfeng-slave 46000000.pfe_slave: Version: 1.4.0
[    0.851109] pfeng-slave 46000000.pfe  slave: Driver commit hash: M4_DRIVER_COMMIT_HASH
[    0.859080] pfeng-slave 46000000.pfe_slave: Multi instance support: SLAVE/mdetect=on
[    0.866960] pfeng-slave 46000000.pfe_slave: Compiled by: 10.2.0
[    0.873156] pfeng-slave 46000000.pfe_slave: Wait for PFE controller UP ...
root@s32g399ardb3:~#
```

8. It needs to cooperate with M-core master drive for verification (to be added in the future).

4 Compiled PFE Master Driver into Kernel

4.1 Compiling the Master driver

Modify the \linux\drivers\net\ethernet\pfe\Kconfig file as follows:

```
…
#when compile the Master driver ,defatult: MASTER=1
config PFE_CFG_PFE_MASTER
        int "PFE master"
        default 1
        help
          If you have a network (Ethernet) card based on PFE IP
          Core

# When compiling a single master driver, you can set MULTI_ INSTANC=0
config  PFE_CFG_MULTI_INSTANCE_SUPPORT
        int "PFE multi instance"
        default 0
        help
          If you have a network (Ethernet) card based on PFE IP
          Core
```

Or used the make menuconfig to reconfigure.

And then re-compile the kernel image.

4.2 Testing

Used the original *. sdcard image, rename pfeng.ko, replace Image, start, and the following error will be reported:

```
5.193137] pfeng 46000000.pfe: Direct firmware load for s32g_pfe_class.fw failed with error -2
[    5.193149] pfeng 46000000.pfe: ERR: (DRIVER) event 1 - Driver runtime error: [pfeng-fw.c:20] Firmware not available: s32g_pfe_class.fw
```

It indicates that the file system is not ready when the PFE master driver loads, and further development is needed to solve the FW loading problem.

4.3 Solve the FW loading fail issue

After compiling the PFE master driver into the kernel, you need to compile the PFE FW into the kernel as follows:

1. Open the compilation macro to compile FW to the kernel:

```
make menuconfig
 |->Device Drivers
 |    |->Generic Driver Options
 |    |    |->Firmware loader
 |    |    |    |->() Build named firmware blobs into the kernel binary      |    | #enter this item and inpu: s32g_pfe_class.fw s32g_pfe_util.fw #FW name
      |    |                  (firmware) Firmware blobs root directory # Note that the default value of this item is/lib/firmware, which refers to the/lib directory of your compiling host. Therefore, it needs to be modified to “firmware”, which will become “firmware” folder under the source code directory of the linux kernel.
```

Saved and checked:

```
vi .config
# Firmware loader
CONFIG_FW_LOADER=y
CONFIG_EXTRA_FIRMWARE="s32g_pfe_class.fw s32g_pfe_util.fw"
CONFIG_EXTRA_FIRMWARE_DIR="firmware"
CONFIG_FW_CACHE=y
# end of Firmware loader
```

Then create a directory “firmware” under the linux source code directory, and put the correct firmware into it:

```
nxa08200@lsv11049:/opt/samba/nxa08200/S32G/BSP37/standalone/linux-pfe$ ls firmware/
s32g_pfe_class.fw  s32g_pfe_util.fw
```

Recompiled.

2. Fix FW check bug：

The following error will be reported after running:

ERR: (DRIVER) event 1 - Driver runtime error: [pfe_pe.c:1860] Unsupported firmware detected: Found revision 1.7.0 (fwAPI:92367c0e25f21f49217a9b08168ad2c8), required fwAPI PFE_CFG_PFE_CT_H_MD5

Related source codes is:

Drvers/net/ethernet/pfe/pfe_pe.c

```
/*  Firmware version check */
                        static const char_t mmap_version_str[] = TOSTRING(PFE_CFG_PFE_CT_H_MD5); // PFE_CFG_PFE_CT_H_MD5 undefined
                        (void)memcpy(
                                        (void*)tmp_mmap,
                                        (const void*)((addr_t)elf_file->pvData + ENDIAN_SW_4B(shdr->sh_offset)),
                                        mmap_size);
                        if(0 != strcmp(mmap_version_str, tmp_mmap->common.version.cthdr))
                        {
                                ret = EINVAL;
                                print_fw_issue(tmp_mmap);
```

So add the follow codes in drivers/net/ethernet/pfe/pfe_platform_cfg.h:

```
#define PFE_CFG_PFE_CT_H_MD5        92367c0e25f21f49217a9b08168ad2c8 //johnli add
```

And then Linux boot successfully with PFE master driver：

3. Boot log of PFE master driver as follows:

```
root@s32g399ardb3:~# dmesg |grep pfe
[    0.000000] OF: reserved mem: initialized node pfebufs@83200000, compatible id shared-dma-pool
[    5.191633] pfeng 46000000.pfe: PFEng ethernet driver loading ...
[    5.191639] pfeng 46000000.pfe: Version: 1.4.0
[    5.191643] pfeng 46000000.pfe: Driver commit hash: M4_DRIVER_COMMIT_HASH
[    5.191646] pfeng 46000000.pfe: Multi instance support: disabled (standalone)
[    5.191650] pfeng 46000000.pfe: Compiled by: 10.2.0
[    5.191671] pfeng 46000000.pfe: Cbus addr 0x46000000 size 0x1000000
[    5.191679] pfeng 46000000.pfe: nxp,fw-class-name: s32g_pfe_class.fw
[    5.191684] pfeng 46000000.pfe: nxp,fw-util-name: s32g_pfe_util.fw
[    5.191723] pfeng 46000000.pfe: netif name: pfe0
```

```
[    5.191730] pfeng 46000000.pfe: DT mac addr: 00:01:be:be:ef:11
[    5.191737] pfeng 46000000.pfe: netif(pfe0) linked phyif: 0
[    5.191742] pfeng 46000000.pfe: netif(pfe0) mode: std
[    5.191754] pfeng 46000000.pfe: netif(pfe0) HIFs: count 1 map 01
[    5.191764] pfeng 46000000.pfe: EMAC0 interface mode: 4
[    5.191849] pfeng 46000000.pfe: netif name: pfe1
[    5.191855] pfeng 46000000.pfe: DT mac addr: 00:01:be:be:ef:22
[    5.191862] pfeng 46000000.pfe: netif(pfe1) linked phyif: 1
[    5.191867] pfeng 46000000.pfe: netif(pfe1) mode: std
[    5.191880] pfeng 46000000.pfe: netif(pfe1) HIFs: count 1 map 02
[    5.191889] pfeng 46000000.pfe: EMAC1 interface mode: 4
[    5.191951] pfeng 46000000.pfe: netif name: pfe2
[    5.191956] pfeng 46000000.pfe: DT mac addr: 00:01:be:be:ef:33
[    5.191963] pfeng 46000000.pfe: netif(pfe2) linked phyif: 2
[    5.191967] pfeng 46000000.pfe: netif(pfe2) mode: std
[    5.191980] pfeng 46000000.pfe: netif(pfe2) HIFs: count 1 map 04
[    5.191989] pfeng 46000000.pfe: EMAC2 interface mode: 10
[    5.192006] pfeng 46000000.pfe: HIF channels mask: 0x0007
[    5.192032] pfeng 46000000.pfe: PFE port coherency enabled, mask 0x1e
[    5.192262] pfeng 46000000.pfe: Clocks: sys=300MHz pe=600MHz
[    5.192276] pfeng 46000000.pfe: Interface selected: EMAC0: 0x4 EMAC1: 0x4 EMAC2: 0xa
[    5.192955] pfeng 46000000.pfe: PFE controller reset done
[    5.193009] pfeng 46000000.pfe: TX clock on EMAC0 for interface sgmii installed
[    5.193043] pfeng 46000000.pfe: RX clock on EMAC0 for interface sgmii installed
[    5.193090] pfeng 46000000.pfe: TX clock on EMAC1 for interface sgmii installed
[    5.193121] pfeng 46000000.pfe: RX clock on EMAC1 for interface sgmii installed
[    5.193180] pfeng 46000000.pfe: TX clock on EMAC2 for interface rgmii-id installed
[    5.193201] pfeng 46000000.pfe: RX clock on EMAC2 for interface rgmii-id installed
[    5.193327] pfeng 46000000.pfe: assigned reserved memory node pfebufs@34000000
[    5.193366] pfeng 46000000.pfe: assigned reserved memory node pfebufs@34080000
[    5.193415] pfeng 46000000.pfe: assigned reserved memory node pfebufs@83200000
[    5.193437] pfeng 46000000.pfe: assigned reserved memory node pfebufs@835e0000
[    5.193510] pfeng 46000000.pfe: Firmware: CLASS s32g_pfe_class.fw [42792 bytes]
[    5.193515] pfeng 46000000.pfe: Firmware: UTIL s32g_pfe_util.fw [20716 bytes]
```

S32G Compiled PFE into Kernel

[5.193526] pfeng 46000000.pfe: PFE CBUS p0x46000000 mapped @ v0xffffffc00b000000 (0x1000000 bytes)

[5.193532] pfeng 46000000.pfe: HW version 0x101

[5.193539] pfeng 46000000.pfe: Silicon S32G3

[5.193546] pfeng 46000000.pfe: Fail-Stop mode disabled

[5.196404] pfeng 46000000.pfe: PFE_ERRORS:Parity instance created

[5.196414] pfeng 46000000.pfe: PFE_ERRORS:Watchdog instance created

[5.196421] pfeng 46000000.pfe: PFE_ERRORS:Bus Error instance created

[5.196426] pfeng 46000000.pfe: PFE_ERRORS:FW Fail Stop instance created

[5.196430] pfeng 46000000.pfe: PFE_ERRORS:Host Fail Stop instance created

[5.196435] pfeng 46000000.pfe: PFE_ERRORS:Fail Stop instance created

[5.196442] pfeng 46000000.pfe: PFE_ERRORS:ECC Err instance created

[5.196451] pfeng 46000000.pfe: BMU1 buffer base: p0xc0000000

[5.196551] pfeng 46000000.pfe: BMU2 buffer base: p0x34000000 (0x80000 bytes)

[5.197811] pfeng 46000000.pfe: register IRQ 87 by name 'PFE BMU IRQ'

[5.197941] pfeng 46000000.pfe: BMU_EMPTY_INT (BMU @ p0x(____ptrval____)). Pool ready.

[5.197950] pfeng 46000000.pfe: BMU_EMPTY_INT (BMU @ p0x(____ptrval____)). Pool ready.

[5.197957] pfeng 46000000.pfe: Firmware .elf detected

[5.197964] pfeng 46000000.pfe: Uploading CLASS firmware

[5.197971] pfeng 46000000.pfe: Selected FW loading OPs to load 8 PEs in parallel

[5.217500] pfeng 46000000.pfe: pfe_ct.h file version"92367c0e25f21f49217a9b08168ad2c8"

[5.232673] pfeng 46000000.pfe: [FW VERSION] 1.7.0, Build: Jun 2 2023, 13:48:57 (nogitaaa), ID: 0x31454650

[5.232952] pfeng 46000000.pfe: EMAC timestamp external mode bitmap: 0

[5.232991] pfeng 46000000.pfe: Uploading UTIL firmware

[5.232995] pfeng 46000000.pfe: Selected FW loading OPs to load 1 PEs in parallel

[5.235418] pfeng 46000000.pfe: pfe_ct.h file version"92367c0e25f21f49217a9b08168ad2c8"

[5.236445] pfeng 46000000.pfe: FW feature: drv_run_on_g3

[5.236450] pfeng 46000000.pfe: FW feature: jumbo_frames

[5.236454] pfeng 46000000.pfe: FW feature: software_vlan_table

[5.236458] pfeng 46000000.pfe: FW feature: timestamping

[5.236462] pfeng 46000000.pfe: FW feature: qos_mapping

[5.236467] pfeng 46000000.pfe: FW feature: core_functionality

[5.236471] pfeng 46000000.pfe: FW feature: extended_features

[5.236475] pfeng 46000000.pfe: FW feature: flexible_router

[5.236480] pfeng 46000000.pfe: FW feature: validate_hif_csum

S32G Compiled PFE into Kernel

[5.236485] pfeng 46000000.pfe: FW feature: err051211_workaround

[5.236491] pfeng 46000000.pfe: FW feature: IPsec

[5.236496] pfeng 46000000.pfe: FW feature: l2_bridge_aging

[5.236501] pfeng 46000000.pfe: FW feature: receive_malformed

[5.236505] pfeng 46000000.pfe: FW feature: ptp_conf_check

[5.236509] pfeng 46000000.pfe: FW feature: vlan_conf_check

[5.236513] pfeng 46000000.pfe: FW feature: hash_load_spread

[5.236517] pfeng 46000000.pfe: FW feature: egress_vlan

[5.236520] pfeng 46000000.pfe: FW feature: ingress_vlan

[5.236524] pfeng 46000000.pfe: FW feature: safety

[5.238903] pfeng 46000000.pfe: VLAN ID incorrect or not set. Using default VLAN ID = 0x01.

[5.238908] pfeng 46000000.pfe: VLAN stats size incorrect or not set. Using default VLAN stats size = 20.

[5.238987] pfeng 46000000.pfe: Software vlan hash table @ p0x20001228

[5.239153] pfeng 46000000.pfe: Fall-back bridge domain @ 0x20000a7c (class)

[5.239158] pfeng 46000000.pfe: Default bridge domain @ 0x20000a74 (class)

[5.240114] pfeng 46000000.pfe: Routing table created, Hash Table @ p0xc00e0000, Pool @ p0xc00e8000 (65536 bytes)

[5.240347] pfeng 46000000.pfe: Feature err051211_workaround: DISABLED

[5.241643] pfeng 46000000.pfe: Failed to set pfe_ts clock. PTP will be disabled.

[5.241650] pfeng 46000000.pfe: MDIO bus 0 disabled: Not found in DT

[5.241655] pfeng 46000000.pfe: MDIO bus 1 disabled: Not found in DT

[5.242520] pfeng 46000000.pfe: MDIO bus 2 enabled

[5.242769] pfeng 46000000.pfe: HIF0 enabled

[5.242972] pfeng 46000000.pfe: HIF1 enabled

[5.243168] pfeng 46000000.pfe: HIF2 enabled

[5.243173] pfeng 46000000.pfe: HIF3 not configured, skipped

[5.243244] pfeng 46000000.pfe pfe0 (uninitialized): Subscribe to HIF0

[5.243251] pfeng 46000000.pfe pfe0 (uninitialized): Host LLTX disabled

[5.243664] pfeng 46000000.pfe pfe0 (uninitialized): Enable HIF0

[5.243824] pfeng 46000000.pfe pfe0 (uninitialized): setting MAC addr: 00:01:be:be:ef:11

[5.244373] pfeng 46000000.pfe pfe0: registered

[5.244396] pfeng 46000000.pfe pfe1 (uninitialized): Subscribe to HIF1

[5.244402] pfeng 46000000.pfe pfe1 (uninitialized): Host LLTX disabled

[5.244788] pfeng 46000000.pfe pfe1 (uninitialized): Enable HIF1

[5.244947] pfeng 46000000.pfe pfe1 (uninitialized): setting MAC addr: 00:01:be:be:ef:22

```
[    5.245342] pfeng 46000000.pfe pfe1: registered
[    5.245358] pfeng 46000000.pfe pfe2 (uninitialized): Subscribe to HIF2
[    5.245363] pfeng 46000000.pfe pfe2 (uninitialized): Host LLTX disabled
[    5.245751] pfeng 46000000.pfe pfe2 (uninitialized): Enable HIF2
[    5.245908] pfeng 46000000.pfe pfe2 (uninitialized): setting MAC addr: 00:01:be:be:ef:33
[    5.246314] pfeng 46000000.pfe pfe2: registered
[    9.111505] pfeng 46000000.pfe: HIF2 started
[   9.179571] pfeng 46000000.pfe pfe2: PHY [PFEng Ethernet MDIO.2:04] driver [Micrel KSZ9031 Gigabit PHY] (irq=POLL)
[    9.179594] pfeng 46000000.pfe pfe2: configuring for phy/rgmii-id link mode
[    9.183490] pfeng 46000000.pfe: HIF1 started
[    9.187704] pfeng 46000000.pfe pfe1: PHY [stmmac-0:08] driver [Aquantia AQR113c] (irq=POLL)
[    9.187722] pfeng 46000000.pfe pfe1: configuring for phy/sgmii link mode
[    9.187799] pfeng 46000000.pfe pfe1: Speed not supported
[    9.193120] pfeng 46000000.pfe: HIF0 started
[    9.193135] pfeng 46000000.pfe pfe0: configuring for fixed/sgmii link mode
[    9.193588] pfeng 46000000.pfe pfe0: Link is Up - 2.5Gbps/Full - flow control off
[    9.193774] IPv6: ADDRCONF(NETDEV_CHANGE): pfe0: link becomes ready
```

ifconfig command as follows:

```
root@s32g399ardb3:~# ifconfig

pfe0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::201:beff:febe:ef11  prefixlen 64  scopeid 0x20<link>
…
        device memory 0x46000000-46ffffff

pfe1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:01:be:be:ef:22  txqueuelen 1000  (Ethernet)
…
        device memory 0x46000000-46ffffff

pfe2: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:01:be:be:ef:33  txqueuelen 1000  (Ethernet)
…
```

device memory 0x46000000-46ffffff