CAS training Rev.2, 7/2023

S32G Git Server Migrate Method

by John Li (nxa08200)

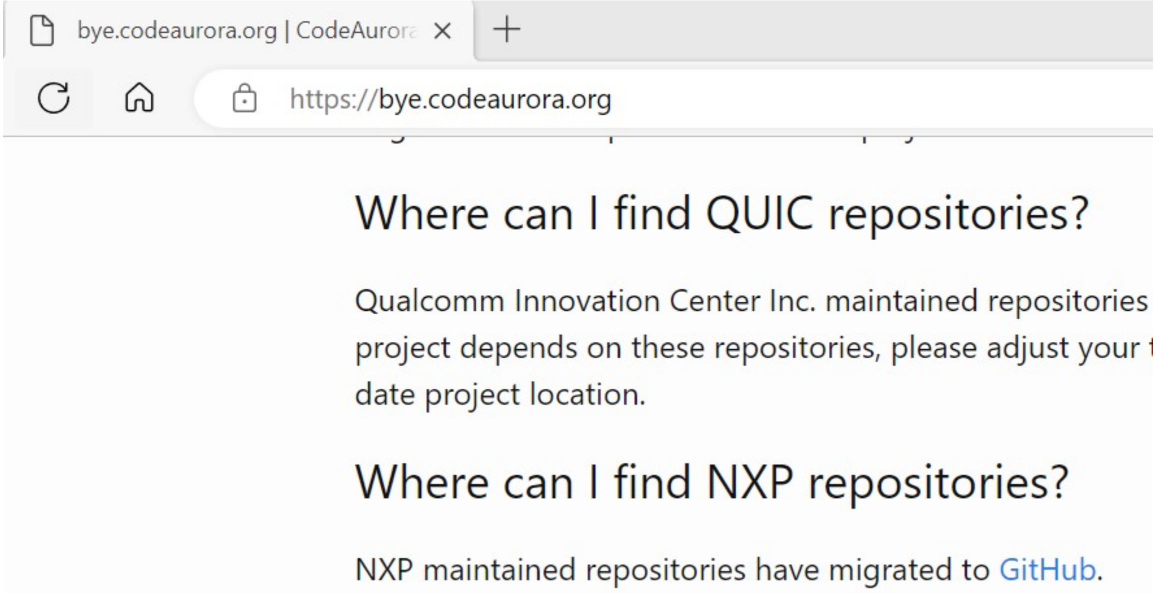
由于 NXP 之前使用的 git 服务器 codeaurora 已经永久性关闭，本文说明针对 S32G 如何迁移到 github 服务器的方法。

历史	说明	作者
V1	● 创建本文	John.Li
V1	● 增加 standalong 编译说明	John.Li

目录

1 服务器迁移情况说明

NXP 所使用的 git 服务器网站 codeaurora 已经于 4 月 1 号起永久性关闭，所以访问 source.codeaurora.org 将失败，参考 codeaurora 的 goodbye 说明https://bye.codeaurora.org/ 如下：



bye.codeaurora.org | CodeAurora

https://bye.codeaurora.org

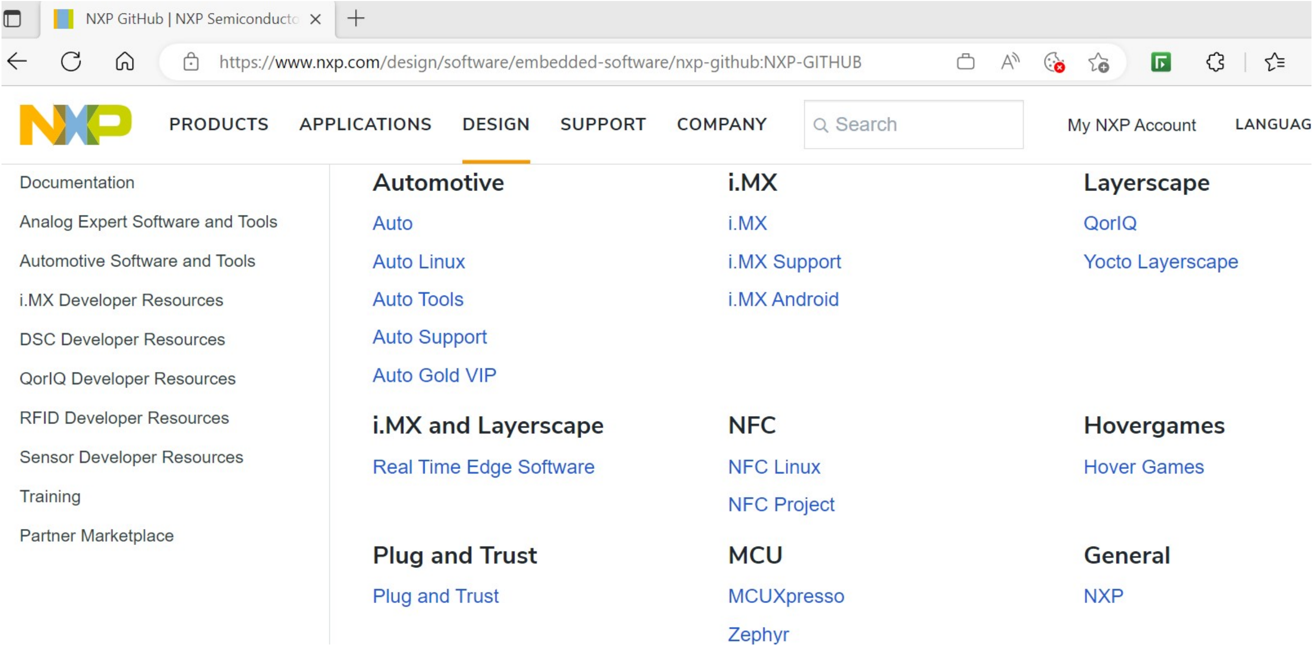
Where can I find QUIC repositories?

Qualcomm Innovation Center Inc. maintained repositories project depends on these repositories, please adjust your t date project location.

Where can I find NXP repositories?

NXP maintained repositories have migrated to GitHub.

点击 NXP repositories 的 GitHub 连接如下：



NXP GitHub | NXP Semiconducto

https://www.nxp.com/design/software/embedded-software/nxp-github:NXP-GITHUB

Documentation
Analog Expert Software and Tools
Automotive Software and Tools
i.MX Developer Resources
DSC Developer Resources
QorIQ Developer Resources
RFID Developer Resources
Sensor Developer Resources
Training
Partner Marketplace

Automotive: Auto, Auto Linux, Auto Tools, Auto Support, Auto Gold VIP

i.MX: i.MX, i.MX Support, i.MX Android

Layerscape: QorIQ, Yocto Layerscape

i.MX and Layerscape: Real Time Edge Software

NFC: NFC Linux, NFC Project

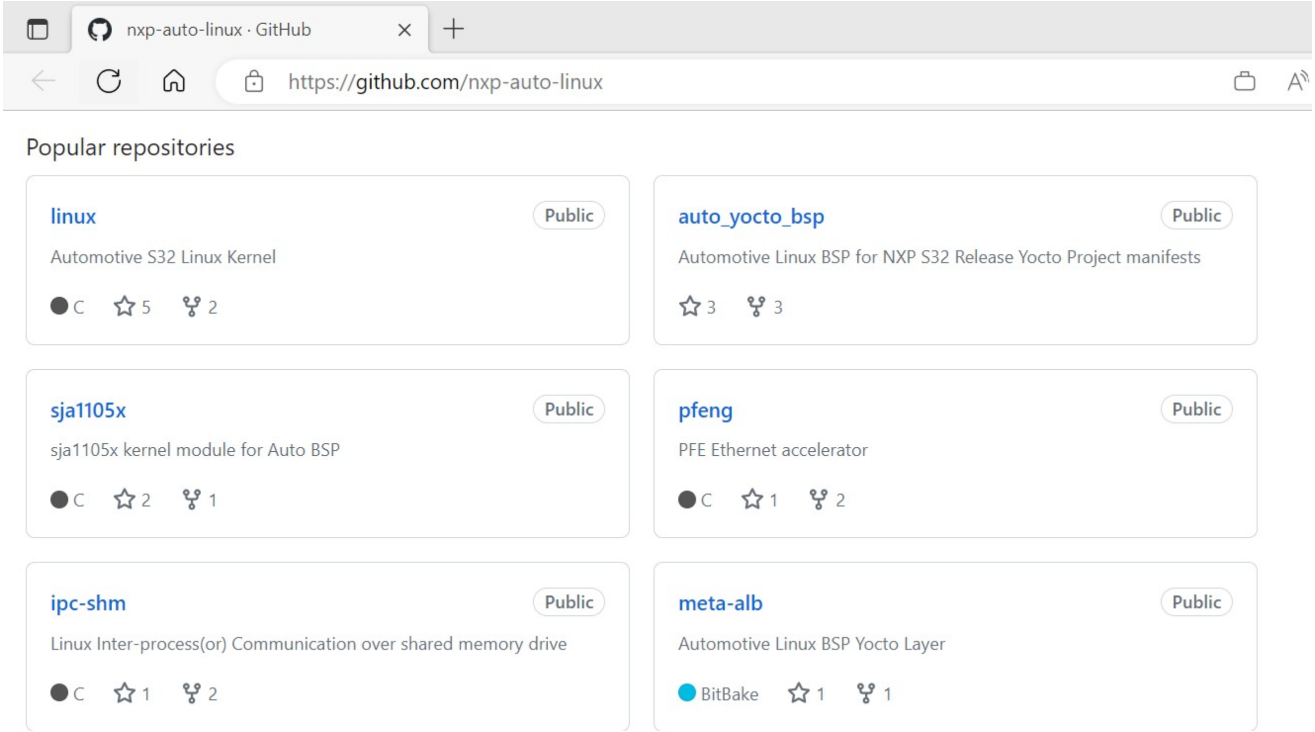
Hovergames: Hover Games

Plug and Trust: Plug and Trust

MCU: MCUXpresso, Zephyr

General: NXP

再点击 Auto Linux 进入到新的 github 网站：



nxp-auto-linux · GitHub

https://github.com/nxp-auto-linux

Popular repositories

linux Public
Automotive S32 Linux Kernel
● C ☆ 5 2

auto_yocto_bsp Public
Automotive Linux BSP for NXP S32 Release Yocto Project manifests
☆ 3 3

sja1105x Public
sja1105x kernel module for Auto BSP
● C ☆ 2 1

pfeng Public
PFE Ethernet accelerator
● C ☆ 1 2

ipc-shm Public
Linux Inter-process(or) Communication over shared memory drive
● C ☆ 1 2

meta-alb Public
Automotive Linux BSP Yocto Layer
● BitBake ☆ 1 1

所以对于之前发布的 S32G 的软件版本，其 Linux 相关文档说明都是基于 codeaurora 的，本文说明对老的版本如何迁移到 github 上。

注意以下迁移策略：

- 从 BSP35 开始(发布时间为 2022.12.22)，已经使用的是 github 服务器了，所以此版本之后的版本不会有问题。
- 对于 BSP34，计划发布 BSP34.0.1 来修改为 github 地址，目前这个发布还没有看到。
- 对老的版本，NXP S32G 提供一个 migrate.sh 脚本，用于 Yocto 的服务器地址迁移的自动修改，本文主要说明此脚本用法。

2 参考文档和工具

本文以 BSP32 为例来说明如何迁移，所以如下文档及软件：

软件/工具/文档	名称	说明
S32G BSP32	● binaries_auto_linux_bsp32.0_s32g2_pfe.tgz ● S32G2_LinuxBSP_32.0_Quick_Start.pdf ● S32G2_LinuxBSP_32.0_Release_Notes.pdf ● S32G2_LinuxBSP_32.0_User_Manual.pdf ● S32G2_LinuxBSP32.0_quality_package.zip ● S32G2_PFE_LinuxBSP32.0_license.txt	从www.nxp.com/s32g 下载
S32G Linux 定制说明	● S32G_ATF_BSP32_V1-20220505.pdf ● S32G_Uboot_BSP32_V5-20220512.pdf ● S32G_Kernel_BSP32_V4-20220513.pdf	FAE写的BSP定制文档，可以从public community里搜索下载
脚本	migrate.sh	Git 服务器地址修改脚本

3 Yocto 迁移说明

Migrate.sh 脚本有两种用法，如下：

3.1 从头开始新建 Yocto 工程说明

全自动模式：在这种模式下，用户从头开始，脚本将处理所有事情，git clone 并将源调整到 NXP GitHub。

首先，请按照 Yocto 编译说明下载 repo 并设置到 PATH 变量：

```
pwd
/opt/samba/nxa08200/S32G/BSP32/bin
nxa08200@lsv11049:/opt/samba/nxa08200/S32G/BSP32/bin$ ls
repo
export PATH=/opt/samba/nxa08200/S32G/BSP32/bin:$PATH
```

然后执行脚本：

```
nxa08200@lsv11049:/opt/samba/nxa08200/S32G/BSP32/test_github$ ./migrate.sh --full --work_path ./testfolder --release_branch release/bsp32.0
```

执行结果如下:

```
=================================================================================
WARNING: THIS SCRIPT WILL CHANGE MANIFEST AND YOCTO RECIPE FILES!
```

```
SINCE CODE AURORA HAS BEEN SHUT DOWN, THIS SCRIPT IS NEEDED IN ORDER
TO CHANGE THE CAF LINKS TO NXP/GITHUB CORRESPONDING LINKS.

IT IS STRONGLY RECOMMENDED THAT USERS PROPERLY SAVE ALL THEIR LOCAL
CHANGES IN REPOSITORES CLONED BY THE 'repo' TOOL BEFORE ATTEMPTING
TO RUN THIS SCRIPT (VIA 'git commit' OR OTHER MEANS)!!!

PROCEEDING FURTHER MEANS ACKNOWLEDGING THE RISKS INVOLVED AND
PROPERLY SAVING ALL YOUR CHANGES!
=================================================================
        Proceed further?(y/n) y
[INFO] Preparing working directory ./testfolder/fsl-image-auto-bsp ...
[INFO] Finished preparing working directory!
[INFO] Performing repo init...
Downloading Repo source from https://gerrit.googlesource.com/git-repo

... A new version of repo (2.32) is available.
... You should upgrade soon:
    cp /opt/samba/nxa08200/S32G/BSP32/test_github/testfolder/fsl-image-auto-bsp/.repo/repo/repo
/opt/samba/nxa08200/S32G/BSP32/bin/repo



Your identity is: yueshu.li@nxp.com <yueshu.li@nxp.com>
If you want to change this, please re-run 'repo init' with --config-name

repo has been initialized in /opt/samba/nxa08200/S32G/BSP32/test_github/testfolder/fsl-image-auto-bsp
/opt/samba/nxa08200/S32G/BSP32/test_github
[INFO] Finished repo init!
[INFO] Starting presync migration!
[INFO] Done presync migration!
[INFO] Performing repo sync...

... A new version of repo (2.32) is available.
... You should upgrade soon:
```

```
cp /opt/samba/nxa08200/S32G/BSP32/test_github/testfolder/fsl-image-auto-bsp/.repo/repo/repo /opt/samba/nxa08200/S32G/BSP32/bin/repo
```

```
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 15 (delta 11), reused 15 (delta 11), pack-reused 0
Unpacking objects: 100% (15/15), done.
Fetching:  0% (0/9) warming up…
Fetching: 100% (9/9), done in 4m32.061s
Checking out: 100% (9/9), done in 1.409s
repo sync has finished successfully.
/opt/samba/nxa08200/S32G/BSP32/test_github
[INFO] Finished repo sync!
[INFO] Starting postsync migration!
[INFO] Done postsync migration!
```

之后就可以直接编译了：

```
source nxp-setup-alb.sh -m <target_machine> //比如：  source nxp-setup-alb.sh -m s32g274ardb2

bitbake <target-image> //比如 bitbake fsl-image-auto
```

3.2 对已有 Yocto 工程的迁移说明

在这种模式下，假设用户已经 clone 了所有仓库，并且在执行“repo sync”后不久遇到了 codeaurora 不可访问的问题警告，则使用此方法。

注意：

在此模式下开始之前，应保存所有本地仓库更改！！！ (例如通过“git add”+“git commit”或其他所需的方法)。

如果之前已经执行了 repo init 下载了源代码，则以以下目录结构为例：

```
pwd
/opt/samba/nxa08200/S32G/BSP32
ls
fsl-auto-yocto-bsp/
mv fsl-auto-yocto-bsp/ fsl-image-auto-bsp  //脚本只接受 fsl-image-auto-bsp 做为目录名，所以需要修改目录名
cd ..
```

```
cp BSP32/test_github/migrate.sh .
./migrate.sh --postsync --work_path ./BSP32
```

所以需要在 BSP32 目录下有 fsl-image-auto-bsp，为之前下载的目录。如下说明：

注意：

work_path 需要一个包含“fsl-image-auto-bsp”子目录的目录。 例如，如果“fsl-image-auto-bsp”在“testfolder”内，我们应该使用相对于它的路径从“testfolder”外部启动脚本。 此外，如果您的“fsl-image-auto-bsp”目录以其他方式命名，您应该运行脚本“migrate.sh”并重命名它。 此限制将在不久的将来解决，以方便使用其他文件夹名称。

执行结果如下：

```
======================================================================
WARNING: THIS SCRIPT WILL CHANGE MANIFEST AND YOCTO RECIPE FILES!
SINCE CODE AURORA HAS BEEN SHUT DOWN, THIS SCRIPT IS NEEDED IN ORDER
TO CHANGE THE CAF LINKS TO NXP/GITHUB CORRESPONDING LINKS.

IT IS STRONGLY RECOMMENDED THAT USERS PROPERLY SAVE ALL THEIR LOCAL
CHANGES IN REPOSITORES CLONED BY THE 'repo' TOOL BEFORE ATTEMPTING
TO RUN THIS SCRIPT (VIA 'git commit' OR OTHER MEANS)!!!

PROCEEDING FURTHER MEANS ACKNOWLEDGING THE RISKS INVOLVED AND
PROPERLY SAVING ALL YOUR CHANGES!
======================================================================
        Proceed further?(y/n) y
[INFO] Starting postsync migration!
[INFO] Done postsync migration!
```

然后就可以正常编译了。

3.3 Migrate.sh 的帮助

帮助说明如下：

```
./migrate.sh --help
Usage:
                    ./migrate.sh OPERATING_MODE [<ARGUMENTS>]
      OPERATING_MODE can be one of the following:
      -> -h | --help
```

```

                shows this help
                Example usage:
                        ./migrate.sh -h
                or
                        ./migrate.sh --help
    ==========================================================================================
    -> -v | --version

                shows the version of ./migrate.sh

                Example usage:
                        ./migrate.sh -v
                or
                        ./migrate.sh --version
    ==========================================================================================
    -> -f | --full

                performs fully-automated handling from scratch
                (clone https://github.com/nxp-auto-linux/auto_yocto_bsp,
                'repo init' and 'repo sync' with URLs adjusted
                to GitHub)
                Mandatory arguments:
                -p | --work_path  desired path for the
                                        fsl-image-auto-bsp directory
                -b | --release_branch   desired upstream
                                        release branch
                Example usage:
                        ./migrate.sh -f -p ./my_folder -b release/bsp33.0
                or
                        ./migrate.sh --full --work_path ./my_folder --release_branch release/bsp33.0
    ==========================================================================================
    -> -s | --sync

                In this mode, we suppose that 'repo init' has been already
                performed, but no 'repo sync' has been performed yet.
                Mandatory arguments:
                -p | --work_path  path for the existing
                                  fsl-image-auto-bsp directory
                Example usage:
                      ./migrate.sh -s -p ./path/to/fsl-image-auto-bsp
                or
                      ./migrate.sh --sync --work_path ./path/to/fsl-image-auto-bsp

   ================================================================================
   -> -ps | --postsync

                In this mode, we suppose that 'repo sync' has been already
                performed before CAF shutdown.

                WARNING:
                Here, we have to be careful to firstly save what we have worked on!
                The script will prompt you to save all your meta-layers repositories
                changes before continuing.
                Mandatory arguments:
                -p | --work_path  path for the existing
                                  fsl-image-auto-bsp directory
                Example usage
                      ./migrate.sh -ps -p /path/to/fsl-image-auto-bsp

                or
                      ./migrate.sh --postsync --work_path /path/to/fsl-image-auto-bsp

   EXTRA / OPTIONAL features:
   -> running in debug mode
                Run in debug mode. Useful for feedback and bug reports.
                Can be used with ANY of the above operationg modes.
                Example usage:
                      DEBUG_MODE=1 ./migrate.sh -ps -p /path/to/fsl-image-auto-bsp
```

```
or
DEBUG_MODE=1 ./migrate.sh --postsync --workpath /path/to/fsl-image-auto-bsp
```

4 独立编译迁移说明

独立编译时比较简单，只需要在 git clone 时修改对应的 git 服务器地址，从 codeaurora 改成 github(https://github.com/nxp-auto-linux/)就可以了，比较 BSP32/BSP35 的 Linux RM 就可得知如下表说明：

软件	BSP32	BSP35
ATF	git clone https://source.codeaurora.org/external/autobsps32/arm-trusted-firmware	git clone https://github.com/nxp-auto-linux/arm-trusted-firmware
Uboot	git clone https://source.codeaurora.org/external/autobsps32/u-boot	git clone https://github.com/nxp-auto-linux/u-boot
Kernel	git clone https://source.codeaurora.org/external/autobsps32/linux	git clone https://github.com/nxp-auto-linux/linux
PFE	git clone https://source.codeaurora.org/external/autobsps32/extra/pfeng/	git clone https://github.com/nxp-auto-linux/pfeng/
IPCF	git clone https://source.codeaurora.org/external/autobsps32/ipcf/ipc-shm/ git clone https://source.codeaurora.org/external/autobsps32/ipcf/ipc-shm-us/	git clone https://github.com/nxp-auto-linux/ipc-shm/ git clone https://github.com/nxp-auto-linux/ipc-shm-us/
Pkcs11-hse	git clone https:// source.codeaurora.org/external/autobsps32/extra/pkcs11-hse	git clone https://github.com/nxp-auto-linux/pkcs11-hse/
Optee_os	git clone https:// source.codeaurora.org/external/autobsps32/extra/optee_os	git clone https://github.com/nxp-auto-linux/optee_os/

注意，在有一些 hard coding 的代码中，特别是在使用 submodule 的方法时，可能会出现写死 codeaurora 地址的情况，比如说以下在 IPCF 代码中：

```
vi ipc-shm-us/.git/config
[submodule "common"]
	active = true
	url = https://github.com/nxp-auto-linux/ipc-shm //change from codeaurora to git hub.
```

此问题已经上报，之后的 IPCF 发布会修复。

注意：

本文基于 BSP32 测试，至于针对其它版本是否可行，还有待验证。

有一些模块是 M/A 都有驱动的情况，所以要注意 MCAL 与 Linux 驱动的版本匹配，如 IPCF 与 PFE，以下说明这些模块独立编译时需要注意的地方，与服务器迁移无关。

4.1 PFE 独立编译

PFE 独立编译需要注意的地方主要是编译器选择，如下以 BSP37(PFE1.4.0)为例：

```
git clone https://github.com/nxp-auto-linux/pfeng/
cd pfeng/
git status
On branch master
Your branch is up to date with 'origin/master'.
git tag

BLN_PFE-DRV_S32G_A53_LNX_1.2.0
BLN_PFE-DRV_S32G_A53_LNX_1.3.0
BLN_PFE-DRV_S32G_A53_LNX_1.4.0
…
TEST-PFE_S32G_A53_LNX_BETA_0.9.0

git checkout BLN_PFE-DRV_S32G_A53_LNX_1.4.0
Note: checking out 'BLN_PFE-DRV_S32G_A53_LNX_1.4.0'.

git status
HEAD detached at BLN_PFE-DRV_S32G_A53_LNX_1.4.0
nothing to commit, working tree clean
```

然后，参考文档 pfeng-1.4.0\doc 《PFE_S32G_A53_LNX_UserManual.pdf》的 **3.1.2 Variant 2: Standalone build** 来完成独立编译：

文中说明如下：

b. PLATFORM
The name of the GNU toolchain platform. In case of Linux BSP, it is “aarch64-linux-gnu”.

所以 standalone 编译时使用的编译器为编译服务器 ubuntu 本身预先安装的 aarch64 编译器 /usr/bin/aarch64-linux-gnu-，而 PFE1.4.0 对应的 BSP37 使用的 ubuntu 版本是 Ubuntu-20.04 LTS，所以在老版本的 ubuntu 编译服务器上，如果使用默认的老版本预先安装的编译器，则可能编译不通过。

另一方面，linux 文档《S32G2_LinuxBSP_37.0_User_Manual.pdf》，中独立编译使用的编译器为：

This Linux BSP has been built and tested using the GCC 11.3.0 for ARM64 toolchain.
The link for the GCC 11.3.0 for ARM64 toolchain, as delivered by Arm at:
https://developer:arm:com/-/media/Files/downloads/gnu/11:3:rel1/binrel/arm-gnu-toolchain-11:3:rel1-x86_64-aarch64-none-linux-gnu:tar:xz?rev=8d05006a68d24d929d602804ec9abfb4&hash=E8A66D3B9FF0EFC60A5C041AC3C5EE476349817B

则如果想使用此编译器编译 PFE 驱动，需要注意以下问题：

```
pwd
~/pfeng/sw/linux-pfeng
vi ../build_env.mak
ifeq ($(TARGET_OS),LINUX)
…
# export PFE_CFG_TARGET_ARCH_DEF=PFE_CFG_TARGET_ARCH_$(shell echo $(PLATFORM) | cut -d '-' -f 1-1)
```

Makefile在判断编译平台时，会将变量PLATFRORM=aarch64-linux-gnu的-号之前的aarch64截取下来，形成正确的平台变量：PFE_CFG_TARGET_ARCH_aarch64。

而同时变量 PLATFRORM 也用于编译器地址指定：

```
pwd
~/pfeng/sw/linux-pfeng
Vi makefile:
… PLATFORM=$(PLATFORM) ARCH=$(ARCH) CROSS_COMPILE=$(PLATFORM)-
```

因为使用 GCC11.3.0 的编译器需要传递绝对地址，所以这儿会出现判断平台失败，可以将 build_env.mak 做如下修改，指定死编译平台：

```
  export PFE_CFG_TARGET_ARCH_DEF=PFE_CFG_TARGET_ARCH_aarch64
```

然后编译命令示例如下：

```
pwd
~/pfeng/sw/linux-pfeng
```

```
make PFE_CFG_MULTI_INSTANCE_SUPPORT=1 PFE_CFG_PFE_MASTER=0
PFE_CFG_TARGET_ARCH_aarch64=1  KERNELDIR=~/BSP37/standalone/linux/
PLATFORM=~/compiler/arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu all
```

假设 GCC11.3.0 安装在~/compiler/arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu/。

如果使用 Yocto SDK 导出安装的编译器则如下：

```
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-linu
x all
```

假设 Yocto SDK 安装在
~/BSP37/fsl-image-auto-bsp/sdk/sysroots/x86_64-fslbsp-linux/usr/bin/aarch64-fsl-linux/aarch64-fsl-linux。

4.2 IPCF 独立编译

IPCF 独立编译需注意 checkout 正确的版本号(以 BSP35, PFE 4.8.0 为例)，首先，确认一下默认 Yocto 中的 IPCF 版本号：

In our bsp35 yocto bb file:

~/BSP35/fsl-auto-yocto-bsp/sources/meta-alb/recipes-kernel/ipc-shm$

```
vi ipc-shm.bb which commitid as follows:
SRCREV = "96f7e0db064cee7a242e2e7289853a7ad9288db8"
```

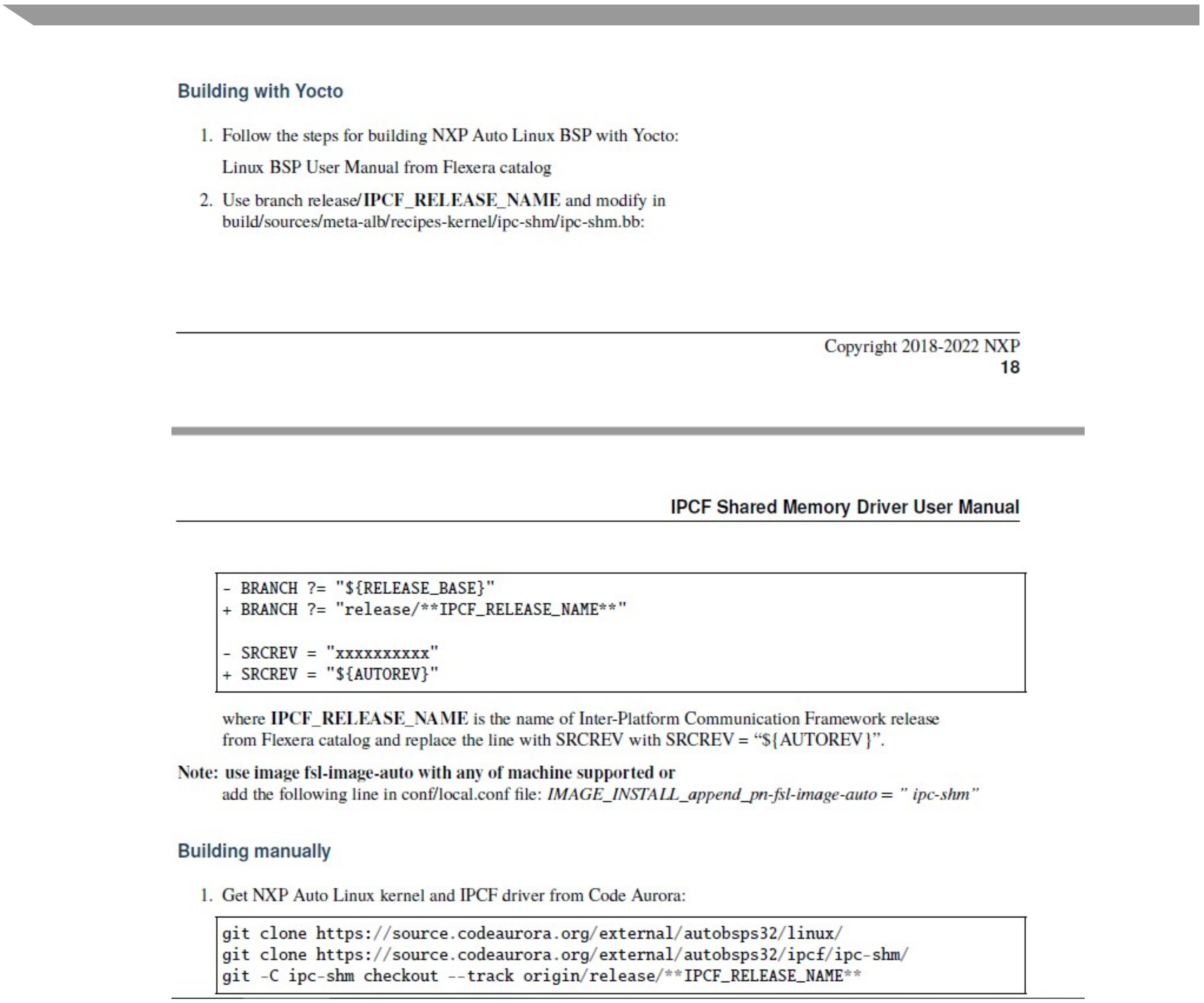
然后在：~/BSP35/standalone/ipcf/ipc-shm$

```
git show 96f7e0db064cee7a242e2e7289853a7ad9288db8
commit 96f7e0db064cee7a242e2e7289853a7ad9288db8
Author: Hoang Nguyen Le <hoang.nguyenle@nxp.com>
Date:   Tue Sep 13 10:00:59 2022 +0700
…
```

所以并没有对应的 tag 号，然后：

```
~/BSP35/standalone/ipcf/ipc-shm$ git show SW32G_IPCF_4.8.0_D2212 -s --format=%H
07321f760bc248f98a1c9ef27ef068ab73640e4b
```

可见虽然 BSP35 和 IPCF_4.8.0 是同一个 bundle release,但是实际上 Yocto 默认编译的驱动版本并不是与 MCAL 匹配的，这一点文档：
C:\NXP\SW32G_IPCF_4.8.0_D2212\eclipse\plugins\Ipcf_TS_T40D11M40I0R0\doc
《UserManual_S32GXXXa.pdf》也有说明，只是说明的比较含糊：



Building with Yocto

1. Follow the steps for building NXP Auto Linux BSP with Yocto:
   Linux BSP User Manual from Flexera catalog
2. Use branch release/**IPCF_RELEASE_NAME** and modify in build/sources/meta-alb/recipes-kernel/ipc-shm/ipc-shm.bb:

Copyright 2018-2022 NXP
18

IPCF Shared Memory Driver User Manual

```
- BRANCH ?= "${RELEASE_BASE}"
+ BRANCH ?= "release/**IPCF_RELEASE_NAME**"

- SRCREV = "xxxxxxxxxx"
+ SRCREV = "${AUTOREV}"
```

where **IPCF_RELEASE_NAME** is the name of Inter-Platform Communication Framework release from Flexera catalog and replace the line with SRCREV with SRCREV = “${AUTOREV}”.

Note: use image fsl-image-auto with any of machine supported or
add the following line in conf/local.conf file: *IMAGE_INSTALL_append_pn-fsl-image-auto = ” ipc-shm”*

Building manually

1. Get NXP Auto Linux kernel and IPCF driver from Code Aurora:

```
git clone https://source.codeaurora.org/external/autobsps32/linux/
git clone https://source.codeaurora.org/external/autobsps32/ipcf/ipc-shm/
git -C ipc-shm checkout --track origin/release/**IPCF_RELEASE_NAME**
```

所以对于 IPCF 驱动，不建议使用 Yocto 默认编译出来的版本，而需要修改 Yocto 配置文件，或者 standalone 编译，并 git checkout 正确的与 MCAL 匹配的版本，如下：

1:

```
git clone https://github.com/nxp-auto-linux/ipc-shm/
cd ipc-shm/
git tag
git checkout SW32G_IPCF_4.8.0_D2212
source ~/BSP35/fsl-auto-yocto-bsp/sdk/environment-setup-cortexa53-crypto-fsl-linux
cd sample
make PLATFORM_FLAVOR=s32g2 KERNELDIR=~/BSP35/standalone/linux/ modules  //G3 use PLATFORM_FLAVOR=s32g3
ls
ipc-shm-sample.ko …
```

```
cd ../sample_multi_instance/
make PLATFORM_FLAVOR=s32g2 KERNELDIR=~/BSP35/standalone/linux/ modules
ls
ipc-shm-sample_multi-instance.ko
```

2:

git clone https://github.com/nxp-auto-linux/ipc-shm-us/

根据前文所说，修改 submodule 的 config git address 的错误。

```
git -C ipc-shm-us submodule update --init –remote
cd ipc-shm-us
git checkout SW32G_IPCF_4.8.0_D2212
cd common/
git checkout SW32G_IPCF_4.8.0_D2212
source ~/BSP35/fsl-auto-yocto-bsp/sdk/environment-setup-cortexa53-crypto-fsl-linux
make PLATFORM_FLAVOR=s32g2 KERNELDIR=/opt/samba/nxa08200/S32G/BSP35/standalone/linux/linux/ modules
ls
ipc-shm-dev.ko   ipc-shm-uio.ko
```

另开一个窗口：

```
cd ../sample
export ARCH=arm64
export CROSS_COMPILE=~/BSP35/fsl-auto-yocto-bsp/sdk/sysroots/x86_64-fslbsp-linux/usr/bin/aarch64-fsl-linux/aarch64-fsl-linux-
export CC="~/BSP35/fsl-auto-yocto-bsp/sdk/sysroots/x86_64-fslbsp-linux/usr/bin/aarch64-fsl-linux/aarch64-fsl-linux-gcc --sysroot=~/BSP35/fsl-auto-yocto-bsp/sdk/sysroots/cortexa53-crypto-fsl-linux/"
make PLATFORM=S32GEN1 IPC_UIO_MODULE_DIR=~/BSP35/fsl-auto-yocto-bsp/sdk/sysroots/cortexa53-crypto-fsl-linux/lib/modules/5.10.90-rt60+g47a49b91d4d4/extra/ CC="$CC"
```

或者：

```
make PLATFORM=S32GEN1 IPC_UIO_MODULE_DIR=../common/ CC="$CC"
ls
ipc-shm-sample.elf
```

这个办法是用的 Yocto 导出的 sdk 的编译器。

其余细节参考：
C:\NXP\SW32G_IPCF_4.8.0_D2212\eclipse\plugins\Ipcf_TS_T40D11M40I0R0\doc《UserManual_S32GXXXa.pdf》。