YNOV

PROJET DE MASTER EN PARTENARIAT AVEC THALES

# Intégration du driver IMX219 sur une OpenRex Basic


*Auteurs :*
Alan AIT-ALI
Martin LAPORTE
Clément AILLOUD
Romain PETIT

*Superviseurs :*
Patrick PIQUART
David COUÉ


*Rapport final présentant le travail effectué sur l'intégration du driver IMX219 sur une OpenRex Basic*

*au sein du*

Département Aéronautique & Systèmes Embarqués


22 février 2018

# Table des matières

# Chapitre 1

# Introduction

## 1.1 Procédure initiale

Cette partie sera une explication des différentes commandes ainsi qu'une description de la méthodologie choisie pour travailler avec Yocto.

### 1.1.1 Gérer son espace mémoire

Tout d'abord sachons que le manuel Yocto conseille de réserver 50 Go afin de pouvoir travailler dans de bonnes conditions. Dans notre cas nous avons une partition de 100 Go incluant Ubuntu14 (5 Go), à terme notre répertoire de travail sera suffisamment agrandi pour combler les 40 Go. Pour ne pas trop encombrer son espace de stockage on peut définir dans build/local.conf la variable

```
1 INHERIT += "rm_work" //effacera les fichiers intermédiaires au fur
     et a mesure
2 RM_OLD_IMAGE = "1" //remplacera les images par la plus récente.
```

Attention ces variables vous feront perdre des données, rm_work par exemple libérera de l'espace (16Go) mais rendra impossible le débogage de la compilation. La suppression des images précédentes est conseillée quand on est amené à en générer beaucoup. Une image pèse quelques centaines de Mo.

### 1.1.2 Préparation de l'environnement

Packages requis pour une compilation Yocto :

- Git 1.8.3.1 ou plus
- Tar 1.27 ou plus
- Python 3.4.0 ou plus

```
user@poky : $ sudo apt-get install gawk wget git-core diffstat unzip texinfo
gcc-multilib  build-essential chrpath socat cpio python python3 python3-pip
python3-pexpect  xz-utils debianutils iputils-ping libsdl1.2-dev xterm repo
```

### 1.1.3 Téléchargement des sources Freescale

La première étape consiste à télécharger tous les exécutables et métadonnées nécessaires pour créer l'image de base de l'OpenRex. En premier lieu il est nécessaire de récupérer la commande repo via les dépots officiels ou téléchargement :

```
sudo aptitude install repo
sudo aptitude update
sudo aptitude upgrade
```

ou

```
mkdir bin
curl http ://commondatastorage.googleapis.com/git-repo-downloads/repo >
bin/repo
chmod a+x bin/repo
PATH=$PATH :bin
```

Ensuite il faut créer le répertoire de travail :

```
mkdir fsl-community-bsp && cd fsl-community-bsp
```

**Yocto version 2.0**

Création du répertoire du fichier manifest :

```
mkdir -p ./repo/local_manifests/
```

Télécharger l'ensemble des sources externes de ce projet par la commande

```
repo init -u https ://github.com/Freescale/fsl-community-bsp-platform -b je-
thro
```

De plus il céer le manifest pour pouvoir télécharger/installer/décompresser les sources produites par Fedevel.

```
cat > .repo/local_manifests/imx6openrex.xml « EOF
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
<remote fetch="git ://github.com/FEDEVEL" name="fedevel"/>
<project remote="fedevel" revision="jethro" name="meta-openrex"
path="sources/meta-openrex">
<copyfile src="openrex-setup.sh" dest="openrex-setup.sh"/>
</project>
</manifest>
EOF
```

Enfin, il est nécessaire de télécharger/installer/décompresser ces fichiers :

```
repo sync
```

**Yocto version 2.1**

```
repo init -u git@github.com/petit-romain/fsl-community-bsp-platform.git
```

Enfin, il est nécessaire de télécharger/installer/décompresser ces fichiers :

```
repo sync
```

### 1.1.4 Compilation ( Yocto 2.0 )

Créer et sourcer l'environnement pour que la compilation inclut la métadonnée Openrex.

```
source openrex-setup.sh
```

Préparer l'environnement et construire l'image. Toutes Les commandes ci-dessous seront à effectuer pendant l'usage courant, elles ne font plus vraiment partie de l'initialisation du projet.

```
MACHINE=imx6s-openrex source setup-environment build-dir
bitbake core-image-base
```

### 1.1.5 Compilation ( Yocto 2.1 )

```
DISTRO=poky MACHINE=imx6-openrexbasic source setup-environment build-
dir bitbake openrexpicam-base-image
```

### 1.1.6 Déploiement de l'image ( Yocto 2.0 )

Installer l'image sur la carte-sd

```
umount /dev/YourSDCard
gunzip -c tmp/deploy/images/imx6s-openrex/core-image-base-imx6s-
openrex.sdcard.gz > tmp/deploy/images/imx6s-openrex/core-image-base-
imx6s-openrex.sdcard
sudo dd if=tmp/deploy/images/imx6s-openrex/core-image-base-imx6q-
openrex.sdcard of=/dev/YourSDCard
umount /dev/YourSDCard
```

A présent insérez la carte sd dans l'OpenRex et démarrez, vous devriez voir l'autoboot apparaître grâce à la communication série.

### 1.1.7 Déploiement de l'image ( Yocto 2.1 )

```
umount /dev/YourSDCard
gunzip -c tmp/deploy/images/imx6-openrexbasic/core-image-base-
imx6-openrexbasic.sdcard.gz > build-openrex/tmp/deploy/images/imx6-
openrexbasic/core-image-base-imx6-openrexbasic.sdcard
sudo dd if=tmp/deploy/images/imx6-openrexbasic/core-image-base-imx6-
openrexbasic.sdcard of=/dev/YourSDCard
umount /dev/YourSDCard
```

Si vous obtenez l'erreur "Error : bootmmc not defined " , redémarrez l'OpenRex et interrompez le boot de celle-ci afin de lancer la commande ci dessous :

```
setenv bootmmc "run findfdt; mmc dev $mmcdev; if mmc rescan; then if run
loadbootscript; then run bootscript; else if run loadimage; then run mmcboot;
else run netboot; fi; fi; else run netboot; fi;\0"; saveenv; reset;
```

### 1.1.8 Etat initial

Une fois la procédure faite, on doit retrouver les fichiers suivant dans le dossier fsl-community-bsp :

fsl-setup-release.sh : script déjà utilisé à la premère préparation de l'environnement.

setup-environment : est utilisé pour sourcer les variables d'environnement utiles lors de la création de l'image.

Build-dir : contient tous les fichiers générés lors des compilations. On y trouve :

bitbake.lock : sémaphore temporaire assurant qu'une seule instance de bitbake accède aux fichiers

sstate-cache : avantage majeur de Yocto sur des compilation standard, dans ce répertoire sont stockées les états de compilation intermédiaire de Yocto. Pour vider la shared state cache on lance :]

```
bitbake <recette> -c cleansstate.
```

tmp : dossier principal de compilation, tmp/work contient les sources des recettes et tmp/deploy les images

sources : contient l'ensemble des métas nécessaires à la création de l'image. On y trouve :]

meta-freescale : configure l'image pour un processeur Freescale.

meta-fsl-arm : configure l'image pour un cortex arm Freescale.

meta-fsl-arm-extra : configure plus encore l'image pour un cortex arm Freescale.

meta-fsl-bsp-release : configure l'image pour une carte officielle à processeur Freescale.

meta-fsl-arm-voipac : surcouche des metas précédentes pour l'Openrex.

meta-openembedded : contient les meta des drivers materiel tel que le bluetooth et les bibliothèques utiles a l'utilisation d'un driver par exemple le python.

meta-fsl-demos : des démonstrations Freescale.

meta-browser : permet d'avoir mozilla ou bien chrome sur l'image.

poky : contient la base du build system et les metadata de base relatif a la distribution.

base : une base pour configurer son dossier build pour les meta Freescale.

meta-qt5 : elle nous permet d'utiliser du code Qt par l'OpenRex en utilisant un SDK. (adresse pour le clonage :https ://github.com/meta-qt5/meta-qt5.git)

### 1.1.9 Ajout de notre méta (Yocto 2.0)

meta-openrexpicam : la méta qui va définir notre distribution. (https ://github.com/Alanaitali/meta-openrexpicam.git) Si vous utilisez Yocto 2.0, notre meta doit simplement être copiée dans le dossier sources puis indiquée dans le bblayers.conf comme expliqué dans « Usage courant » ci-dessous. (follow me)

```
cd $BUILDDIR/../sources/. git clone https ://github.com/Alanaitali/meta-openrexpicam.git
```

## 1.2 Usage courant

Préparation de l'environnement L'environnement de développement a été créé avec fsl-setup-release.sh, mais pour chaque terminal de travail, il faudra sourcer l'environnement à nouveau. C'est-à-dire ajouter des variables d'environnement qui seront nécessaire lors de la compilation. La commande suivante source l'environnement et dans le même temps nous lui indiquons le nom de la machine dans notre cas « imx6s-Openrex » que l'on retrouve dans les sources téléchargées. De cette manière nous donnons des paramètres qui seront pris en compte lors de la compilation avec le « bitbake ». Cette assignation de machine sera inscrite dans le local.conf de manière plus pérenne.

**Yocto 2.0 :**

```
MACHINE=imx6s-openrex source setup-environment build-dir bitbake core-image-base
```

**Yocto 2.1 :**

```
DISTRO=poky MACHINE=imx6-openrexbasic source setup-environment build-dir
```

### 1.2.1 Vérification de l'image demandée

Avant de lancer la compilation nous devons indiquer le chemin d'accès et le nom des métadonnées à prendre en compte pour la compilation. Ces paquets sont aussi appelés couches (layers), ils sont référencés dans le fichier bblayer.conf dans le dossier /build-dir/conf. Pour inclure une nouvelle meta à la compilation il faut écrire son chemin sur bblayers.conf.

```
1  POKY_BBLAYERS_CONF_VERSION = "2"
2
3  BBPATH = "\${TOPDIR}"
4  BSPDIR := "\${@os.path.abspath(os.path.dirname(d.getVar('FILE', 
       True)) + '/../..')}"
5  BBFILES ?= ""
6
7  BBLAYERS = " \
8  \${BSPDIR}/sources/poky/meta \
9  \${BSPDIR}/sources/poky/meta-poky \
10 \
11 \${BSPDIR}/sources/meta-openembedded/meta-oe \
12 \${BSPDIR}/sources/meta-openembedded/meta-multimedia \
13 \
14 \${BSPDIR}/sources/meta-fsl-arm \
15 \${BSPDIR}/sources/meta-fsl-arm-extra \
16 \${BSPDIR}/sources/meta-fsl-demos \
17 \
18 \${BSPDIR}/sources/meta-openrexpicam \
19 "
20 BBLAYERS += "\${BSPDIR}/sources/meta-fsl-arm-voipac"
```

Afin de configurer l'environnement de travail de bitbake on rédige un fichier local.conf. Attention à ce que le nom de machine (imx6-openrexbasic pour Yocto 2.1 et imx6s-openrex) soit en accord avec la recette utilisée.

```
MACHINE ??= 'imx6-openrexbasic'
DISTRO ?= 'poky'
PACKAGE_CLASSES ?= "package_rpm"
EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
USER_CLASSES ?= "buildstats␣image-mklibs"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
STOPTASKS,\${TMPDIR},1G,100K␣\
STOPTASKS,\${DL_DIR},1G,100K␣\
STOPTASKS,\${SSTATE_DIR},1G,100K␣\
STOPTASKS,/tmp,100M,100K␣\
ABORT,\${TMPDIR},100M,1K␣\
ABORT,\${DL_DIR},100M,1K␣\
ABORT,\${SSTATE_DIR},100M,1K␣\
ABORT,/tmp,10M,1K"
PACKAGECONFIG_append_pn-qemu-native = "␣sdl"
PACKAGECONFIG_append_pn-nativesdk-qemu = "␣sdl"
CONF_VERSION = "1"

DL_DIR ?= "\${BSPDIR}/downloads/"
ACCEPT_FSL_EULA = "1"
```

Par soucis de personnalisation la distribution du projet peut être renommée et le nombre de threads configurés en ajoutant :

```
DISTRO ?= 'name'
BB_NUMBER_THREADS?="4"
PARALLEL_MAKE?="-j4"
```

### 1.2.2 Compilation

L'étape compilation prend bien moins de temps lors de l'usage régulier (près de 24 fois plus rapide). Pendant cette étape nous avons un grand nombre d'informations qui s'affichent sur le terminal, les plus importantes sont les warnings et les erreurs de compilation qu'il est nécessaire de comprendre puis interpréter pour résoudre.

```
bitbake openrexpicam-base-image
```

Une fois cette étape terminée on peut retrouver toutes une arborescence de dossier et de fichiers dans le dossier de construction (/build-dir), ces derniers seront au cœur du fonctionnement de notre appareils. ils sont spécifiques à notre carte, les modules, les capteurs qu'elle comporte. Le fruit de la compilation de l'image contient le root-filesystem le kernel le bootloader et le device-tree en un seul fichier compressé. /build-dir/tmp/deploy/images/imx6-openrexbasic doit contenir :

- 1 image sous format .sdcard.gz
- 1 image sous format .rootfs.ext4
- 1 image sous format .rootfs.manifest

- 1 image sous format .rootfs.sdcard