

Y.N.O.V.

PROJET DE MASTER EN PARTENARIAT AVEC THALÈS

Intégration d'un driver sur une OpenRex Basic

Auteurs :

Alain AIT-ALI
Martin LAPORTE
Clément AILLOUD
Romain PETIT

Superviseurs :

Patrick PIQUART
David COUÉ

*Rapport hebdomadaire présentant l'avancement du projet sur l'intégration d'un driver de la
Raspi Cam v2 sur un imx6 OpenRex Basic*

au sein du

Département Aéronautique & Systèmes Embarqués

23 janvier 2018

Table des matières

1	Organisation de l'équipe et planning	1
1.1	Organisation de l'équipe	1
1.2	Planning	1
2	Piste n° 1	3
2.1	Erreur	3
2.2	Solution	4
2.3	Conclusion	4
2.4	Erreur	4
2.5	Solution	5
2.6	Conclusion	5
2.7	Erreur	6
2.8	Reste à faire	6
3	Solution n° 2	7
3.1	Kernel 4.14	7
3.2	Travail effectué	7
3.3	Erreur	8
3.4	Conclusion	8
3.5	Reste à faire	8
4	Compilation d'un OS avec un kernel 3.14	9
4.1	Commande repo	9
4.1.1	Gestionnaire de paquet	9
4.1.2	Téléchargement	9
4.2	Préparation du dépôt	9
4.2.1	Création du dépôt	9
4.2.2	Récupération des sources	9
4.3	Utilisation de notre meta	10
4.3.1	Téléchargement	10
4.3.2	Implémentation	10
4.4	Compilation	10
5	Conclusion	11

Table des figures

1.1	Avancement général du projet	1
1.2	Avancement détaillé du projet	2
2.1	Chargement du module	3
2.2	Debug probe platform data	3
2.3	Debug probe clk	4
2.4	Chargement du module	5
2.5	Dmsg	6
3.1	Arborescence des fichiers	7
3.2	Erreur du bootloader	8

Chapitre 1

Organisation de l'équipe et planning

1.1 Organisation de l'équipe

Afin de se concentrer sur la "Solution n°2" lundi 8 nous nous sommes lancés dans l'analyse du code c du driver et de son fonctionnement. Le code réparti entre chacun, nous avons cherché à comprendre les utilités des structures et de l'organisation d'un client kernel. Nous avons donc chacun poursuivit les appels aux fichiers systemes, en mutualisant les informations oralement.

Grâce à cette organisation agile nous avons pu préciser la source du problème avant d'en chercher la solution.

Par la suite Romain suivi par Alan se sont lancés dans le second objectif d'évolution, la compilation d'un noyau en version 4.14. Ce afin de précéder le portage du driver à ce kernel.

1.2 Planning

Ci-dessous, un planning des tâches effectuées en 2018.

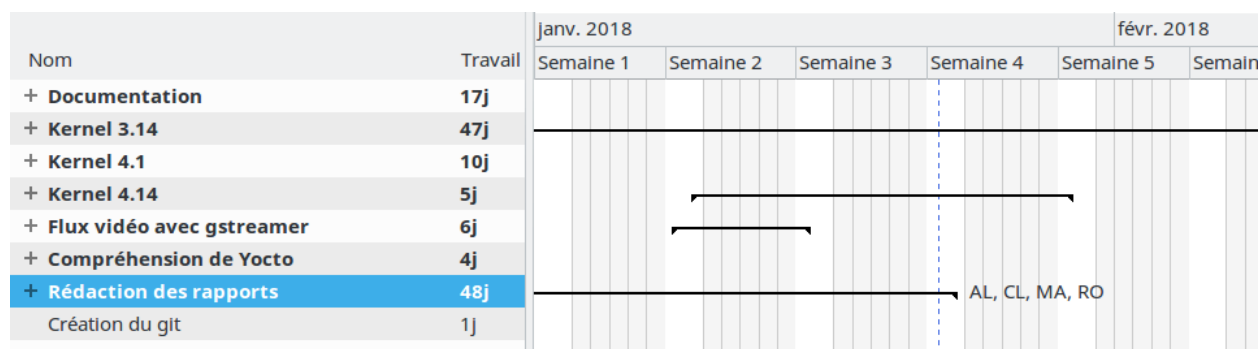


FIGURE 1.1 – Avancement général du projet

AL = Alan Ait-Ali

MA = Martin LAPORTE

RO = Romain Petit

CL = Clément Ailloud

La totalité du planning est accessible sur notre git, sur la branche "presentation" à l'emplacement "meta-openrexpica/presentation2/ganthales.planner" sous forme de fichier à ouvrir avec le logiciel "planner" (sudo aptitude install planner).

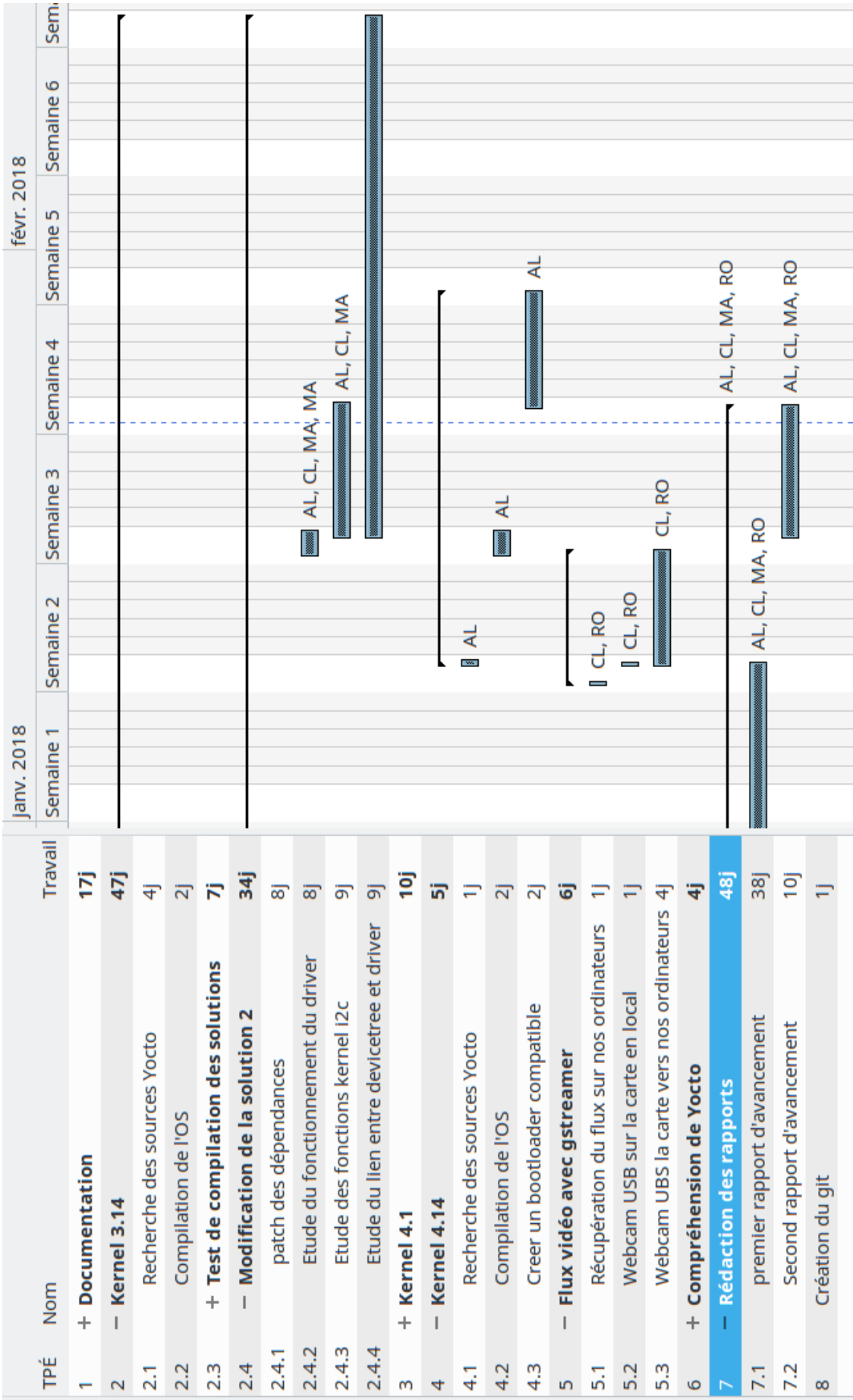


FIGURE 1.2 – Avancement détaillé du projet

Chapitre 2

Piste n° 1

Version du Kernel : 3.14

Code de travail : imx219.c (fichier de la solution 2 sur le rapport du 09/01/2017)

Suite à l'exploration des 3 pistes, nous avons choisi de nous concentrer sur la solution n°2 puisque le driver se compile sans erreur. Cependant il ne se charge pas dans le kernel.

2.1 Erreur

Constat : Le driver apparait à la commande lsmod mais est inutilisé.

```
root@openrexpica:~# modprobe imx219
root@openrexpica:~# lsmod
Module                  Size  Used by
mxc_v4l2_capture        25109  0
ipu_bg_overlay_sdc      5242   1 mxc_v4l2_capture
ipu_still               2312   1 mxc_v4l2_capture
ipu_prp_enc            5872   1 mxc_v4l2_capture
ipu_csi_enc            3743   1 mxc_v4l2_capture
v4l2_int_device         2913   2 ipu_csi_enc,mxc_v4l2_capture
ipu_fg_overlay_sdc      6068   1 mxc_v4l2_capture
imx219                  7888   0
mxc_dcic                6543   0
galcore                225000  0
evbug                  1871   0
```

FIGURE 2.1 – Chargement du module

Après recherche d'indices pour le debug, on obtient le message d'erreur suivant :

```
imx219 1-0064: IMX219: missing platform data!
```

FIGURE 2.2 – Debug probe platform data

Résultat de l'analyse : l'erreur provient de la fonction probe du driver. La sécurité testant le premier paramètre d'appel du driver s'est déclenchée. Le paramètre est une structure contenant des informations sur les périphériques i2c. Il y a donc un problème de compatibilité entre le driver imx219 et le gestionnaire i2c. Selon l'avis de notre professeur de linux embarqué qui a appuyé nos recherches, le driver en question utilise les interfaces de fonctionnement nommées platform_data, technologie remplacée progressivement depuis 2011 par le device-tree et sa gestion des compatibilités.

2.2 Solution

Aujourd'hui mardi 16 janvier nous avons cherché à porter le driver vers une compatibilité avec le device-tree.

Ajout de la structure type of_device_id dans le driver

```
1 static const struct of_device_id imx219_of_match[] = {
2     { .compatible = "sony,imx219", .data = 0 },
3     {}
4 };
5
6 MODULE_DEVICE_TABLE(of, imx219_of_match);
```

Completion de la structure de manipulation du driver

```
1 static struct i2c_driver imx219_i2c_driver = {
2     .driver = {
3         .name = "imx219",
4         .of_match_table = of_match_ptr(imx219_of_match),
5     },
6     .probe = imx219_probe,
7     .remove = imx219_remove,
8     .id_table = imx219_id,
9 };
```

Suppression de la sécurité platform data

```
1 if (!ssdd) {
2     dev_err(&client->dev, "IMX219:_missing_platform_data!\n");
3     return -EINVAL;
4 }
```

2.3 Conclusion

Une nouvelle erreur est apparue cependant les travaux qui concernent le device tree ont fonctionné correctement car on peut remarquer au démarrage que le driver est chargé automatiquement.

2.4 Erreur

Au chargement de la nouvelle image nous obtenons l'erreur suivante :

```
imx219 1-0064: Error -19 getting clock
i2c 1-0064: Driver imx219 requests probe deferral
```

FIGURE 2.3 – Debug probe clk

L'erreur est relative à la fonction getclk du driver qui est une sécurité comme la fonction supprimée précédemment. Elle confirme que nous avons des difficultés à communiquer avec l'i2c.

2.5 Solution

Notre stratégie est de supprimer cette nouvelle fonction de sécurité, pour avoir plus d'informations sur la façon dont le driver fonctionne.

Suppression de la sécurité d'horloge

```

1      if (IS_ERR(priv->clk)) {
2          dev_info(&client->dev, "Error_%ld_getting_clock\n",
3              PTR_ERR(priv->clk));
4          return -EPROBE_DEFER;
5      }

```

2.6 Conclusion

Le driver est maintenant chargé "correctement" par le kernel.

```

root@openrexpica:~# i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- 48 -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- UU -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --
root@openrexpica:~# lsmod
Module                Size  Used by
mxc_v4l2_capture      25109  0
ipu_bg_overlay_sdc    5242   1 mxc_v4l2_capture
ipu_still             2312   1 mxc_v4l2_capture
ipu_prp_enc           5872   1 mxc_v4l2_capture
ipu_csi_enc           3743   1 mxc_v4l2_capture
v4l2_int_device       2913   2 ipu_csi_enc,mxc_v4l2_capture
ipu_fg_overlay_sdc    6068   1 mxc_v4l2_capture
imx219                7716   1
mxc_dcic              6543   0
galcore              225000  0
evbug                1871   0
root@openrexpica:~#

```

FIGURE 2.4 – Chargement du module

On peut voir que le driver imx219 est chargé par l'i2c et qu'il est utilisé par un autre module. Nous ne sommes toujours pas en mesure de nous en servir car il nous manque des informations sur son fonctionnement. De plus de nouvelles erreurs apparaissent au chargement du driver.

2.7 Erreur

Au chargement du driver nous obtenons le message de debug kernel suivant :

```
Modules linked in: ipu_prp_enc ipu_csi_enc v4l2_int_device ipu_fg_overlay_sdc imx219(+) mxc_dcic galcore(0+) evbu
g
CPU: 0 PID: 156 Comm: udevd Tainted: G          0 3.14.61-yocto+g336bc38 #1
task: 863a2d00 ti: 864ae000 task.ti: 864ae000
PC is at soc_camera_power_init+0x0/0xc
LR is at imx219_probe+0xa8/0x424 [imx219]
pc : [<804a09b0>]   lr : [<7f053b0c>]   psr: 80010013
sp : 864afd88 ip : 864afc4d fp : 7f054680
r10: 86164220 r9 : 00000000 r8 : 86164200
r7 : 00000000 r6 : 00000000 r5 : 8655ce10 r4 : 7f053f48
r3 : 8655ce30 r2 : 00000020 r1 : 00000000 r0 : 86164220
Flags: Nzcv IRQs on FIQs on Mode SVC_32 ISA ARM Segment user
Control: 10c53c7d Table: 164b0059 DAC: 00000015
Process udevd (pid: 156, stack limit = 0x864ae238)
Stack: (0x864afd88 to 0x864b0000)
fd80: 8684edc8 80af0698 86164220 86164200 7f05451c 00000005
fda0: 00000000 80af0698 86164220 00000000 7f05451c 00000000 00000000 00000001
fdc0: 7f054680 80353e10 86164220 7f05451c 86164254 80a6b248 8684fd48 80354014
fde0: 00000000 7f05451c 80353f88 8035248c 860c185c 8614cb34 7f05451c 8643a180
fe00: 00000000 80353590 7f054400 7f056000 7f054644 7f05451c 7f056000 7f054644
fe20: 7f054638 8035460c 7f0544f8 7f056000 7f054644 80463cc4 864ae000 800088a0
fe40: 8bc278c0 10000000 8bc278c0 80a25598 00000000 80a95a80 00000000 800aa54c
fe60: 00012e80 800d85bc 00000001 800cedc8 00000093 00000004 00012e84 800d85bc
fe80: 00000001 8008b080 8684fd40 864aff58 00000001 864aff58 00000001 7f054644
fea0: 7f054638 8684fd48 8684fd40 8008b0b8 7f054644 00007fff 800885b4 864d0b40
fec0: 00000000 864aff58 864ae030 80088750 a0d65d6b 00000000 0000007c 80708710
fee0: a0d63000 00003780 000000b1 00000000 0b300002 00000000 00000000 00000000
ff00: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
ff20: 00000000 00000000 00000000 00000000 0000001c 00000000 00000005 76f11dc4
ff40: 0000017b 8000e5a4 864ae000 00000000 01665a48 8008ba38 a0d63000 00003780
ff60: a0d66280 a0d64f67 a0d65870 00001790 00001b90 00000000 00000000 00000000
ff80: 0000001e 0000001f 00000018 00000015 00000013 00000000 00000000 0164d008
ffa0: 00000000 8000e420 00000000 0164d008 00000005 76f11dc4 00000000 00000000
ffc0: 00000000 0164d008 00000000 0000017b 0164d2d8 00000000 0002ea98 01665a48
ffe0: 7ea18c00 7ea18bf0 76f0ac30 76e93760 60010010 00000005 1bf5e821 1bf5ec21
[<804a09b0>] (soc_camera_power_init) from [<7f053b0c>] (imx219_probe+0xa8/0x424 [imx219])
[<7f053b0c>] (imx219_probe [imx219]) from [<80353e10>] (driver_probe_device+0x118/0x24c)
[<80353e10>] (driver_probe_device) from [<80354014>] (__driver_attach+0x8c/0x90)
[<80354014>] (__driver_attach) from [<8035248c>] (bus_for_each_dev+0x68/0x9c)
[<8035248c>] (bus_for_each_dev) from [<80353590>] (bus_add_driver+0x148/0x1f0)
[<80353590>] (bus_add_driver) from [<8035460c>] (driver_register+0x78/0xf8)
[<8035460c>] (driver_register) from [<80463cc4>] (i2c_register_driver+0x30/0xb8)
[<80463cc4>] (i2c_register_driver) from [<800088a0>] (do_one_initcall+0xfc/0x158)
[<800088a0>] (do_one_initcall) from [<8008b0b8>] (load_module+0x1794/0x1f90)
[<8008b0b8>] (load_module) from [<8008ba38>] (SyS_finit_module+0x80/0x90)
[<8008ba38>] (SyS_finit_module) from [<8000e420>] (ret_fast_syscall+0x0/0x38)
Code: e1a00007 e3481092 ebfac115 eaffffe1 (e5912024)
V4L2 device 'Mxc Camera' on IPU1_CSI0 registered as video0
V4L2 device 'Mxc Camera' on IPU1_CSI1 registered as video1
ERROR: v4l2 capture: slave not found!
ERROR: v4l2 capture: slave not found!
---[ end trace faab8b2921e8f95b ]---
udev[152]: worker [156] terminated by signal 11 (Segmentation fault)
udev[152]: worker [156] failed while handling '/devices/soc0/soc.0/2100000.aips-bus/21a4000.i2c/i2c-1/1-0064'
```

FIGURE 2.5 – Dmsg

2.8 Reste à faire

- Déterminer la cause du segmentation fault.
- Savoir comment récupérer le flux video.

Chapitre 3

Solution n° 2

Version du Kernel : 4.14

3.1 Kernel 4.14

La version 4.14 du kernel contient un driver imx219 main-line opérationnel. Notre idée était de porter la meta-voipac compatible avec un kernel 4.1 sur un kernel 4.14 pour pouvoir utiliser le driver.

3.2 Travail effectué

Modifier les sources du Kernel

Dans les recettes chaque version du kernel est architecturée de la même façon :

- un fichier linux-voipac_4.14.bb
- un dossier linux-voipac-4.14 contenant un defconfig et un patch du device tree de l'Openrex

```
walker@walker-OMEN-by-HP-Laptop:~/Yocto/fsl-community-bsp/sources/meta-fsl-arm-voipac/recipes-kernel/
linux$ tree
.
├── linux-voipac-3.14
│   └── defconfig
├── linux-voipac_3.14.bb
├── linux-voipac-4.1
│   └── defconfig
├── linux-voipac-4.14
│   ├── 0001-imx6s-6q-add-initial-support.patch
│   └── defconfig
├── linux-voipac_4.14.bb
└── linux-voipac_4.1.bb
```

FIGURE 3.1 – Arborescence des fichiers

Le principe de cette méthode est de copier la version 4.1 et de remplacer les sources du kernel.

Modification du fichier

```
1 SRCBRANCH = "4.14.x+fslc"
2 LOCALVERSION = "-yocto"
3 SRCREV = "${AUTOREV}"
4 KERNEL_SRC ?=
    "git://github.com/Freescale/linux-fslc.git;protocol=git"
```

3.3 Erreur

Nous obtenons quelques erreurs relative à gstreamer et alsa pendant la compilation du nouveau kernel. Ces paquets n'étant pas essentiels à l'utilisation du driver nous les avons simplement enlevé les paquets.

L'étape de compilation réussite, nous obtenons une erreur lors du boot :

```
Out:  serial
Err:  serial
Net:  FEC
Normal Boot
Hit any key to stop autoboot:  0
## Error: "findfdt" not defined
switch to partitions #0, OK
mmc0 is current device
reading boot-imx6-openrexbasic.scr
** Unable to read file boot-imx6-openrexbasic.scr **
reading zImage
6533744 bytes read in 312 ms (20 MiB/s)
Booting from mmc ...
reading imx6-openrexbasic.dtb
39861 bytes read in 18 ms (2.1 MiB/s)
Kernel image @ 0x10800000 [ 0x000000 - 0x63b270 ]
## Flattened Device Tree blob at 18000000
   Booting using the fdt blob at 0x18000000
   Using Device Tree in place at 18000000, end 1800cbb4

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
```

FIGURE 3.2 – Erreur du bootloader

3.4 Conclusion

U-boot n'arrive pas à lancer notre image, il y a donc une incompatibilité entre la version du kernel et celle du bootloader chargée dans la SPI flash. Pour parvenir à utiliser cette nouvelle image nous devons concevoir un nouveau bootlaoder en suivant la méthode présentée ici.

3.5 Reste à faire

- Créer un nouveau bootlaoder.
- Tester le nouveau bootlaoder.

Chapitre 4

Compilation d'un OS avec un kernel 3.14

4.1 Commande repo

4.1.1 Gestionnaire de paquet

```
$ : sudo aptitude install repo
```

```
$ : sudo aptitude update
```

```
$ : sudo aptitude upgrade
```

4.1.2 Téléchargement

```
$ : mkdir bin
```

```
$ : curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > bin/repo
```

```
$ : chmod a+x bin/repo
```

```
$ : PATH=$PATH:bin
```

4.2 Préparation du dépôt

4.2.1 Création du dépôt

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

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

4.2.2 Récupération des sources

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

```
$ : cat > .repo/local_manifests/imx6-openrexbasic.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
$ : repo sync
$ : source openrex-setup.sh
```

4.3 Utilisation de notre meta

La meta que nous avons créée doit se placer dans le dossier source.

4.3.1 Téléchargement

```
$: cd sources
```

```
$: git clone git@github.com :Alanaitali/meta-openrexpica.git
```

4.3.2 Implémentation

```
$: cd ..
```

```
$: MACHINE=imx6s-openrexpica source setup-environment build-dir
```

```
$: bitbake-layers add-layer ../sources/meta-openrexpica
```

4.4 Compilation

```
$: MACHINE=imx6s-openrexpica bitbake openrexpica-base-image
```

Chapitre 5

Conclusion

De part l'avancée que nous avons actuellement et l'expérience accumulée au cours du projet il nous reste plusieurs pistes à explorer. Sans contre- indication particulière nous devons être capables d'avoir des résultats dans quelques semaines. Nous ne sommes pas en capacité de dire avec précision à quelle date il sera possible de vous fournir un driver fonctionnel.

Pour tout conseil, piste à suivre ou erreur de notre part n'hésitez pas à revenir vers nous. De plus nous tenons à nous excuser pour le dernier rapport envoyé il n'était manifestement pas du tout adapté à votre demande.