Compilation croisée

Table des matières

1 Installation du système cible

Nous allons dans cette partie récupérer l'environnement de développement fournit avec le Kit de développement :

- Compilation de la chaîne de développement croisée
- Compilation du noyau
- Installation du noyau et du système de fichier sur la cible
- Programmation en C et débogage

```
$ mount /media/cdrom
$ cp /media/cdrom/CPUAT91_20.tar.bz2 .
$ tar -jxf CPUAT91_20.tar.bz2
```

1.1 Mise en place des outils de développement

```
# apt-get install gcc flex bison libgmp3-dev libncurses5-dev
```

Attention, bien que les log se plaignent de ne pas avoir libexpat1-dev, il ne faut pas l'installer sinon cela génère une erreur concernant zlib1g-dev lors de la compilation de dropbear à l'étape (partie) suivante.

```
$ ./build_tools.sh
```

Ce script construit les principaux utilistaires (crosstool, gdb, mkfs.jffs2, mkcramfs, loader, sx) :

- Si pour chacun d'eux, il ne trouve pas la place d'un marqueur dans le répertoire *stage* alors,
- il copie les sources depuis le répertoire *PKG_CPUAT91*
- dans l'espace de compilation : *build.* **(écrasement)**
- les résultats de la compilation sont visible dans le répertoire *log*
- et les executables seront pour la majeur partie rangés dans le répertoire *tools* et accessoirement *images*

1.2 Bootloader et noyau

```
# apt-get install libacl1-dev
```

Remarque, le paquet ZLIB1G-DEV semble être facultatif.

```
$ ./build_distrib.sh
```

De façon trés similaire au script précédent, celui-ci construit les composantes de notre OS :

- Si pour chacune d'elles, il ne trouve pas la place d'un marqueur dans le répertoire *stage* alors,
- il copie les sources depuis le répertoire *PKG_CPUAT91*
- dans l'espace de compilation : *build*. **(écrasement)**
- les résultats de la compilation sont visible dans le répertoire *log*
- et les executables seront pour la majeur partie rangés dans le répertoire *rootfs* et accessoirement *images*.
- efin, le script crée les images compressées du système de fichier. (ce que fait aussi le script *build_image.sh*).

Voici les composantes :

- uboot
  - *images/u-boot_ram.bin* chargé par le script *flash_cpuat91.sh*
  - *tools/gcc-4.1.2-glibc-2.3.6/arm-linux/bin/mkimage*
- linux
- rootfs
- busybox
- gdb
- gdbserver
- strace
- eeprog
- gpio
- dosfs
- devmem2
- cpuat91
- libs (anl BrokenLocale c crypt dl gcc_s m memusage nsl nss_compat nss_dns nss_files nss_hesiod nss_nis nss_nisplus pcprofile pthread resolv rt SegFault stdc++ thread_db util zlib)
- modules

1.3 Installation Linux From Scratch

La procédure suivante permet de réinstaller notre OS. Elle constera grosso-mod à :

- Charger le programme (`sx-at91 -s /dev/ttyS0 images/loader.bin`)
- Charger le shell utilisé par la suite (`sx-at91 -s /dev/ttyS0 images/u-boot_ram.bin`)
- Charger le bootloader, le noyau et l'image du système de fichier.
  - */tftpboot/u-boot_cpuat91.bin*
  - */tftpboot/uImage_cpuat91*
  - */tftpboot/rootfs_cpuat91.jffs2*

Relier le port série du poste client au port **UART DBG** au moyen d'un câble série croisé (dit femelle - femelle). Positionner les Jumpers bleu **Jmp9** et **Jmp10** afin de connecter le port série **UART DBG**. **Connecter** le jumper rouge **Jmp11**.

Pensez à appuyer sur le bouton reset (le bouton jaune 'RST').

```
$ cp images/u-boot_cpuat91.bin images/uImage_cpuat91 images/rootfs_cpuat91.jffs2 /tftpboot
$ ./flash_cpuat91.sh /dev/ttyS0
```

Une fois U-BOOT flashé, on reconnecte le terminal série. **Remarque :** l'adresse mac entrée ci-dessous est facultative.

```
$ minicom

CPUAT91=> help
CPUAT91=> ping 192.168.1.1
Micrel KS8721 PHY detected : 100BT FD
host 192.168.1.1 is alive
CPUAT91=> ethaddr 0a:03:4a:c6:f3:a2
...

setenv ipaddr 192.168.1.2;
setenv serverip 192.168.1.1;
saveenv

tftp 20000000 u-boot_cpuat91.bin;
protect off 10000000 1001FFFF;
erase 10000000 1001FFFF;
cp.b 20000000 10000000 ${filesize};
protect on 10000000 1001FFFF

tftp 20000000 uImage_cpuat91;
protect off 10040000 1019FFFF;
erase 10040000 1019FFFF;
cp.b 20000000 10040000 ${filesize}

tftp 20000000 rootfs_cpuat91.jffs2;
protect off 101A0000 107FFFFF;
erase 101A0000 107FFFFF;
cp.b 20000000 101A0000 ${filesize}

setenv bootargs root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS0,115200 \
mtdparts=physmap-flash.0:256k(u-boot),1408k(kernel),-(rootfs) mem=32M
```

```
setenv bootcmd bootm 10040000;
setenv bootdelay 1;
saveenv
```

Résumé : (pour copier/coller) :

- setenv ipaddr 192.168.1.2 ; setenv serverip 192.168.1.1 ; saveenv
- tftp 20000000 u-boot_cpuat91.bin ; protect off 10000000 1001FFFF ; erase 10000000 1001FFFF ; cp.b 20000000 10000000 ${filesize} ; protect on 10000000 1001FFFF
- tftp 20000000 uImage_cpuat91 ; protect off 10040000 1019FFFF ; erase 10040000 1019FFFF ; cp.b 20000000 10040000 ${filesize}
- tftp 20000000 rootfs_cpuat91.jffs2 ; protect off 101A0000 107FFFFF ; erase 101A0000 107FFFFF ; cp.b 20000000 101A0000 ${filesize}
- setenv bootargs root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS0,115200 mtdparts=physmap-flash.0 :256k(u-boot),1408k(kernel),-(rootfs) mem=32M ; setenv bootcmd bootm 10040000 ; setenv bootdelay 1 ; saveenv

cf (on sait jamais) `#define CONFIG_EXTRA_ENV_SETTINGS` dans *build/u-boot-1.2.0/include/configs/cpuat91.h*

Pour tester, retirer le jumper rouge **Jmp11** et générez un reset en pressant le bouton.

Via le port série :

```
eukrea  login: root
Password: eukrea
login[864]: root login on 'ttyS0'
```

Via SSH :

```
$ ssh root@192.168.1.2 -p 2222
root@192.168.1.2's password: eukrea
#
```

1.4 Développement C

Sur le PC hôte :

```
./setup_env.sh
Configuration de l'environnement pour la compilation croisée
Vous êtes à présent dans l'environnement de cross-compilation
Tapper CTRL+D pour restaurer l'environnement initial2

$ cd exemples/
$ make
```

Sur la carte :

```
# cd /tmp
# tftp -g -r test 192.168.1.1
# chmod +x test
# ./test
Hello World !
i = 15 - j = 12

# gdbserver 192.168.1.5:2345 ./test
```

A nouveau sur le PC hôte :

```
$ arm-linux-gdb test_g

(gdb) target remote 192.168.1.2:2345
...
```

Afin de faire communiquer la carte nous nous intéresserons, dans la partie suivante, à la programmation des GPIO.

2 Modification du système de fichier

Il s'agit d'une sous partie de l'algorithme exposé ci-dessus. Le script *rebuild_distrib.sh* va construire pas à pas le système de fichier. Il sera :

- Décompressé depuis *PKG_CPUAT91/rootfstree-cpuat91.tar.bz2* **(écrasement)**
- Enrichi des composantes *busybox devmem2 dosfs dropbear eeprog gpio gdbserver libs rootfs strace zlib modules et cpuat91.*
- Compréssé en 2 images : *rootfs_cpuat91.cramfs* et *rootfs_cpuat91.jffs2* la première aboutissant à un système de fichier en lecture seule tandis que la seconde permet de modifier directement la mémoire flash bien que toutefois les logs soient enregistrées en RAM afin de l'économiser.

Modifier l'adresse mac temporaire (non requis) :

```
$ ./build/u-boot-1.2.0/tools/gen_eth_addr
1a:94:65:70:e0:bf
```

Nous allons à présent modifier l'archive contenant le système de fichier utilisé.

```
$ cd PKG_CPUAT91
$ mkdir tmp
$ cd tmp
$ tar -xjf ../rootfstree-cpuat91.tar.bz2
```

fichier *etc/network/interface* :

```
# Configure Loopback
auto lo eth0
iface lo inet loopback

iface eth0 inet static
       address 192.168.1.2
       netmask 255.255.255.0
       gateway 192.168.1.1
```

fichier *etc/hostname* :

```
eukrea
```

fichier *etc/hosts* :

```
127.0.0.1       localhost
```

fichier *etc/issue* :

```
 -----------------
( Agate the bouze )
 -----------------
        o   ^__^
         o  (xx)\\_______
            (__)\\       )\\/\\
             U  ||----w |
                ||     ||
```

Il faut refermer l'archive contenant le système de fichier puis reconstruire l'image.

```
$ tar -cjf ../rootfstree-cpuat91.tar.bz2 .
$ cd ../..
$ ./rebuild_distrib.sh
$ cp images/* /tftpboot/
```

Le script *rebuild_distrib.sh* invoque le script *build_distrib.sh* décrit ci-dessus aprés avoir effacé les stages *busybox devmem2 dosfs dropbear eeprog gpio gdbserver libs rootfs strace zlib modules et cpuat91.*

A tester : configuration du réseau via DHCP.

3 Modification du noyau linux

Il faut suivre manuellement les instruction du script *build_distrib.sh.* En effet, le script n'aura aucun effet si les fichiers 'stages' ne sont pas détruits. En revanche s'il le sont, les sources seront écrasées avec toute vos modifications.

Certaines GPIO ne sont pas utilisables tel que (y compris en retirant les jumper). Il faut libérer les GPIO relatives aux UARTS 0, 1 et 3 dans le fichier *build/linux-2.6.22.1/arch/arm/mach-at91/board-cpuat91.c* :

```
/*
 * Serial port configuration.
 *    0 .. 3 = USART0 .. USART3
 *    4      = DBGUG
 */
static struct at91_uart_config __initdata cpuat91_uart_config = {
.console_tty = 0,/* ttyS0 */
.nr_tty = 5,
.tty_map = { 4, 2} /* ttyS0, ..., ttyS4 */
};
```

Recompilation et installation :

```
$ cd build/linux-2.6.22.1
$ make menuconfig

$ vi arch/arm/mach-at91/board-cpuat91.c
$ vi arch/arm/mach-at91/at91rm9200_devices.c

$ make uImage modules
$ make INSTALL_MOD_PATH=../../rootfs/ modules_install
$ cp arch/arm/boot/uImage ../../images/uImage_cpuat91
$ cd ../..
$ ./build_images.sh
$ cp images/* /tftpboot/
```

Réinstallez linux comme décrit deux paragraphes plus haut.