EMBEDDED LINUX

Add DCAN driver support to the Linux kernel

Brief
The CAN is not supported by our current Linux configuration. We'll edit the kernel `.config` file in order to enable the CAN driver support as a built-in service, following the Texas Instruments documentation. We'll then deploy the new kernel image onto the SD card to validate this step.

Author
dimitri.boudier@ensicaen.fr
hugo.descoubes@ensicaen.fr

Resources
Processor SDK Linux Software Developer's Guide

Reminder
/!\ Understand all commands before running them! /!\

KERNEL CONFIGURATION FOR SUPPORTING THE DCAN DRIVER

In this section, you will edit the kernel configuration so that Linux contains support for the DCAN driver for the AM335x processor.
Prior to doing anything, let's save the original `.config` file, i.e. the one we've been using from the beginning.

```
cd ${DISCOPATH}/kernel/bb-kernel/KERNEL
cp .config ../patches/.config_original
```

The TI documentation provides information about how to configure the kernel in order to make the DCAN driver functionnal for the AM335x processor : Processor SDK Linux Software Developer's Guide.

In section `3.2.1.4. Configuring the Kernel` it is suggested to start with a default configuration. Let's use the `defconfig` file, which is supplied by Robert C. Nelson, and see the default configuration lines for the CAN device.

```
cp ../patches/defconfig .config
cat .config | grep 'CAN'
```

- Write this below.

Now browse the documentation and find the page dedicated to the DCAN kernel driver (`3.2.4.4. DCAN`). *(Note: if the page is unreachable, a backup PDF is to be found in `disco/docs`).* Then you should find the `Detailed Kernel Configuration` section.

- You can copy-paste here the required configuration.

Apply the required configuration on the `.config` file. Note that we will configure all required services as **built-in** (and not as **modules**). Don't forget to save afterwards.

```
make ARCH=arm menuconfig
```

Verify new `.config` file configuration and see applied changes.

```
grep 'CAN' .config
```

- Note the differences with the original `.config` file.

Save the `.config` file that now contains CAN support.

```
cp .config ../patches/.config_can
```

BUILD KERNEL WITH NEW BUILT-IN SERVICES

In the chapter `3.2.1. Users Guide` of its documentation, TI gives the steps needed for the kernel image compilation. It is suggested to start by cleaning the kernel sources.

Look at the `make` help menu.

```
make ARCH=arm help
```

- What will `make ARCH=arm distclean` do?

See old generated files (`.config`, `.dtb` and `zImage`).

```
ls -la
ls arch/arm/boot/dts | grep .dtb
ls -l arch/arm/boot
```

Before rebuilding the whole kernel image, we clean thekernel file system.

```
make ARCH=arm distclean
```

Verify after cleaning

```
ls -la
ls arch/arm/boot/dts | grep .dtb
ls -l arch/arm/boot
```

- Which files have disappeared?

See, we deleted the `.config` file we just generated. Hopefully we made a backup! Let's bring it back.

```
cp ../patches/.config_can .config
```

Now we can build a new kernel image, which will contain DCAN support.

```
make ARCH=arm CROSS_COMPILE=${CC} zImage -j16
```

- Note that the `-j16` option does not appear in the TI documentation. What's its effect?

Wait until the build is complete (it should takes few minutes). The last line of the building feed shows the Linux image file location:

> Kernel: arch/arm/boot/zImage is ready

Copy this image to our `deploy` directory.

```
cp arch/arm/boot/zImage ${DISCOPATH}/deploy/zImage_can
```

The new kernel image with CAN support is now ready to be deployed.

DEPLOY NEW KERNEL

You will now deploy the new kernel image. Make sure you have a full Debian distribution on your SD card before going on. Otherwise run your `sdcard_deploy.sh` script.

Deploy the kernel image onto the SD card.

```
cd ${DISCOPATH}/deploy
sudo cp -v ./zImage_can ${MEDIA}/rootfs/boot/vmlinuz-${kernel_version}
sync
sudo umount ${MEDIA}/rootfs
```

Start the BeagleBone Black, log in and verify that the CAN driver is now supported by the kernel.

```
dmesg | grep 'CAN\|can:'
```

```
root@arm:~# dmesg | grep 'CAN|can:'
[    1.227598] CAN device driver interface
[    1.622052] can: controller area network core (rev 20170425 abi 9)
[    1.632835] can: raw protocol (rev 20170425)
[    1.637129] can: broadcast manager protocol (rev 20170425 t)
[    1.642827] can: netlink gateway (rev 20170425) max_hops=1
```

- What does `dmesg` do? What's the meaning of the output just above?

Like Ethernet and Wi-Fi, the CAN protocol is usually recognised as a network by Linux. Verify that the CAN network is supported by the kernel.

```
ifconfig -a
```

- Which interfaces do you see? Is the CAN interface visible?

As it stands, the Linux kernel integrates the driver but cannot use it as network. We will see how to change that on the next chapter using Device Tree support!