Adding Wi-Fi Support to iMX28evk

1 Introduction

This document explains how to add Wi-Fi capabilities to the iMX28evk using Yocto. The Wi-Fi module to be used is the AR6103 of Atheros.

2 Requirements

Basic knowledge of Yocto and Linux is required. The below training found in the communities cover the required knowledge to understand this document:

https://community.freescale.com/docs/DOC-94849

It is assumed that the previously mentioned training was followed and that the environment is set up already.

3 Procedure

The general steps to get the AR6103 up and running are the next ones:

- Adding wireless packages to the rootfs. The mainline kernel will be used in this case. The version of the kernel is 3.14.1.
- Adding the atheros firmware to the rootfs.
- Generate and install the needed modules (.ko files) into the rootfs.
- Modify the device tree to configure the second SD slot that is used for the AR6103 module.

3.1 Generating and adding wireless package

To use the mainline kernel for the iMX28evk, to add the wireless package and to populate the .ko files in the rootfs , the local.conf files needs to be changed and the below lines have to be added.

To choose the mainline kernel:
PREFERRED_PROVIDER_virtual/kernel = "linux-fslc"

To add the Atheros and wireless firmware:
IMAGE_INSTALL_append = " linux-firmware iw"

To populate the modules in the rootfs:
MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS += "kernel-modules"

3.2 Configuring the Kernel

The kernel has to be configured to generate and add the modules for the AR6103. This is accomplished by performing the next command.

bitbake -c menuconfig linux-fslc

After thi,s the menuconfig window shows up and the next configurations must be enabled:

- **Networking_support->Wireless->cfg80211 - wireless configuration API**
- **Networking_support->Wireless->cfg80211 wireless extensions compability**

The next configurations are added as modularized features (**M**):

- **Device Drivers->Network device support->Wireless LAN->Atheros Wireless Cards-> Atheros mobile chipsets support**
- **Device Drivers->Network device support->Wireless LAN->Atheros Wireless Cards-> Atheros ath6kl SDIO support**

At this point you can bitbake your image and deploy your SD card. In this case the core-image-minimal is used.

bitbake core-image-minimal

To deploy the SD card you can follow this link:
https://community.freescale.com/docs/DOC-94989

3.3 Modifying the device tree

The iMX28evk shares the pins of the second SD slot for the NAND flash. By default these pins are configured to work with the NAND flash. Therefore the pin configuration has to be changed as well as adding the second node in the device tree to enable the SD slot.
The device tree file is found at:

arch/arm/boot/dts/

In the imx28-evk.dts the next nodes are added:

```
ssp1: ssp@80012000 {
        compatible = "fsl,imx28-mmc";
        pinctrl-names = "default";
        pinctrl-0 = <&mmc1_8bit_pins_a
                &mmc1_cd_cfg &mmc1_sck_cfg>;
        bus-width = <8>;
```

```
                wp-gpios = <&gpio0 28 0>;
                vmmc-supply = <&reg_vddio_sd1>;
                status = "okay";
            };


    reg_vddio_sd1: vddio-sd1 {
            compatible = "regulator-fixed";
            regulator-name = "vddio-sd1";
            regulator-min-microvolt = <3300000>;
            regulator-max-microvolt = <3300000>;
            gpio = <&gpio3 29 0>;
    };
```

The gpmi node has to be removed:

```
            gpmi-nand@8000c000 {
                pinctrl-names = "default";
                pinctrl-0 = <&gpmi_pins_a &gpmi_status_cfg
                             &gpmi_pins_evk>;
                status = "okay";
            };
```

In the imx28.dtsi file the needed pins are added:

```
                mmc1_8bit_pins_a: mmc1-8bit@0 {
                    reg = <0>;
                    fsl,pinmux-ids = <
                        MX28_PAD_GPMI_D00__SSP1_D0
                        MX28_PAD_GPMI_D01__SSP1_D1
                        MX28_PAD_GPMI_D02__SSP1_D2
                        MX28_PAD_GPMI_D03__SSP1_D3
                        MX28_PAD_GPMI_D04__SSP1_D4
                        MX28_PAD_GPMI_D05__SSP1_D5
                        MX28_PAD_GPMI_D06__SSP1_D6
                        MX28_PAD_GPMI_D07__SSP1_D7
                        MX28_PAD_GPMI_RDY1__SSP1_CMD
                        MX28_PAD_GPMI_RDY0__SSP1_CARD_DETECT
                        MX28_PAD_GPMI_WRN__SSP1_SCK
                    >;
                    fsl,drive-strength = <MXS_DRIVE_8mA>;
                    fsl,voltage = <MXS_VOLTAGE_HIGH>;
                    fsl,pull-up = <MXS_PULL_ENABLE>;
                };
                mmc1_cd_cfg: mmc1-cd-cfg {
                    fsl,pinmux-ids = <
                MX28_PAD_GPMI_RDY0__SSP1_CARD_DETECT
        >;
        fsl,pull-up = <MXS_PULL_DISABLE>;
};

mmc1_sck_cfg: mmc1-sck-cfg {
        fsl,pinmux-ids = <
                MX28_PAD_GPMI_WRN__SSP1_SCK
        >;
        fsl,drive-strength = <MXS_DRIVE_12mA>;
        fsl,pull-up = <MXS_PULL_DISABLE>;
};
```

After this change we have to compile and get the .dtb file that needs to be passed to the kernel. To do so we can use the meta-toolchain. The below links show how to generate and use the toolchain

https://community.freescale.com/docs/DOC-95122
https://community.freescale.com/docs/DOC-95225

As shown in the above links, we have to perform the below steps after creating the toolchain:

- Run the environment.sh **source /opt/poky/1.6/environment-setup-armv5te-poky-linux-gnueabi**
- **ARCH=arm**
- **CROSS_COMPILE=$TARGET_PREFIX**
- **unset LDFLAGS**
- **make mxs_defconfig**
- **make imx28-evk.dts**

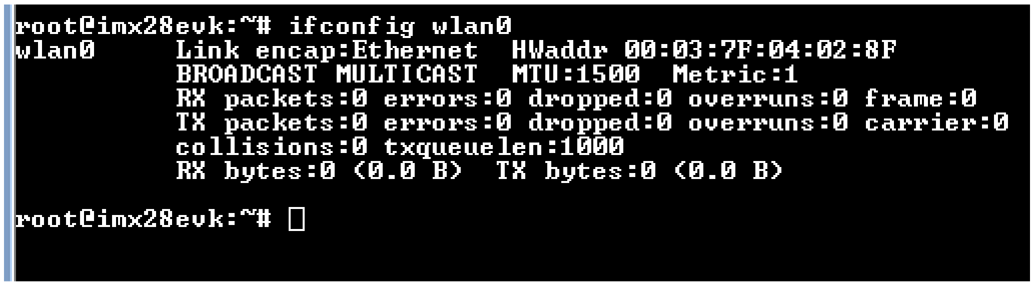
This will generate a .dtb file (imx28-evk.dtb) that needs to be copied into the SD card FAT partition.

3.4 Connecting to a Wireless LAN

Now you can boot up the board. After booting the board, you have to install the .ko files:

```
insmod /lib/modules/3.14.1-fslc+g387df1b/kernel/drivers/net/wireless/ath/ath6kl/ath6kl_core.ko
insmod /lib/modules/3.14.1-fslc+g387df1b/kernel/drivers/net/wireless/ath/ath6kl/ath6kl_sdio.ko
```

Now you have to plug in the Wi-Fi card AR6103.
If you type **ifconfig wlan0** you should see something like:



```
root@imx28evk:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:03:7F:04:02:8F
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@imx28evk:~# 
```

To bring the interface up:
ifconfig wlan0 up

To check the available access points:

iw dev wlan0 scan

Provide an IP to the module.

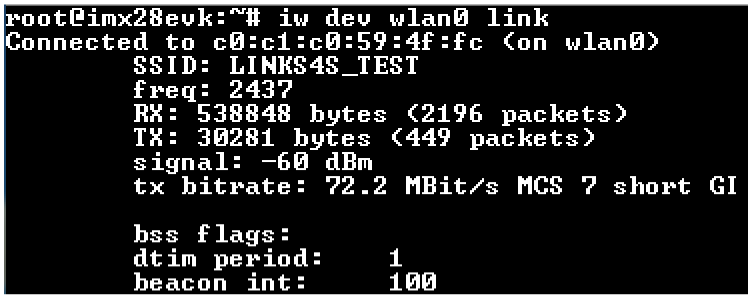
ifconfig wlan0 192.168.1.136

To associate of wireless device to Access Point with no encryption is as follows: (LINKS4S_TEST SSID)

iw dev wlan0 connect LINKS4S_TEST

To make sure you are connected you can type:

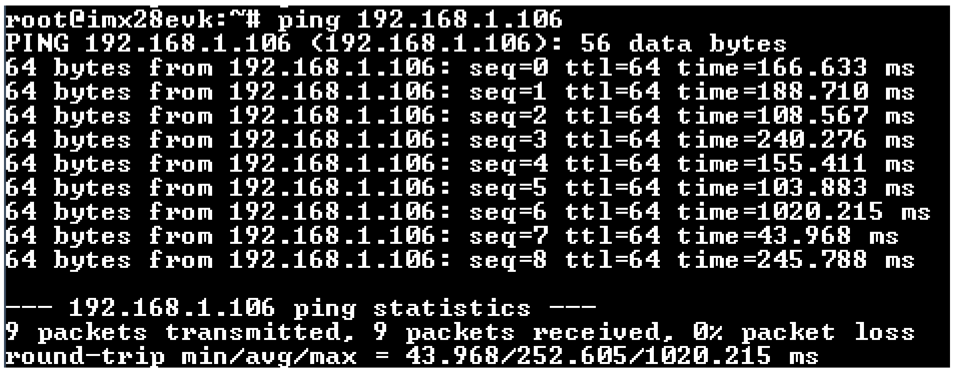
iw dev wlan0 link



```
root@imx28evk:~# iw dev wlan0 link
Connected to c0:c1:c0:59:4f:fc (on wlan0)
        SSID: LINKS4S_TEST
        freq: 2437
        RX: 538848 bytes (2196 packets)
        TX: 30281 bytes (449 packets)
        signal: -60 dBm
        tx bitrate: 72.2 MBit/s MCS 7 short GI

        bss flags:
        dtim period:    1
        beacon int:     100
```

By this time you can try to ping other node in the network.



```
root@imx28evk:~# ping 192.168.1.106
PING 192.168.1.106 (192.168.1.106): 56 data bytes
64 bytes from 192.168.1.106: seq=0 ttl=64 time=166.633 ms
64 bytes from 192.168.1.106: seq=1 ttl=64 time=188.710 ms
64 bytes from 192.168.1.106: seq=2 ttl=64 time=108.567 ms
64 bytes from 192.168.1.106: seq=3 ttl=64 time=240.276 ms
64 bytes from 192.168.1.106: seq=4 ttl=64 time=155.411 ms
64 bytes from 192.168.1.106: seq=5 ttl=64 time=103.883 ms
64 bytes from 192.168.1.106: seq=6 ttl=64 time=1020.215 ms
64 bytes from 192.168.1.106: seq=7 ttl=64 time=43.968 ms
64 bytes from 192.168.1.106: seq=8 ttl=64 time=245.788 ms

--- 192.168.1.106 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 43.968/252.605/1020.215 ms
```