Support ADC in the Linux i.MX RT1170 BSP

Detailed Requirements and Design

rm6903-drad-1_0.doc

RM:	*6903*
Revision:	*1.0*
Date:	*11/14/2023*

TABLE OF CONTENTS

1. Overview

The following is a high-level overview of the problem being resolved by this project:

This project develops Linux ADC device driver in the Linux BSP for the i.MX RT1170 processor.

2. Requirements

2.1. Detailed Requirements

The following are the requirements for this project:

1. Provide a Linux demo project combining all the requirements in this project.
    - *Rationale*: Needed to let Customer integrate results of this project into target embedded application.
      Implementation: Section: "Design: Demo project".
      Test: Section: "Test Plan: Demo Project".

2. Develop Linux ADC device driver for the i.MX RT1170 ADC controller.

    - *Rationale*: Explicit Customer requirement.
      Implementation: Section: "Design: Linux ADC Device Driver".
      Test: Section: "Test Plan: Linux ADC Driver".

2.2. Detailed Non-Requirements

The following are the non-requirements for this project that may otherwise not be obvious:

- None

3. Design

3.1. Detailed Design

3.1.1. Design: Demo project

This project will enable the required ADC functionality in the Linux configuration ("embedded project") called `rootfs`, which resides in a `projects/rootfs` directory, relative to the top of the Linux i.MX RT1170 installation.

3.1.2. Design: Linux ADC Device Driver

The clock driver for IMXRT1170 SoC will be updated to support the ADC clock gates.

There is the existing `imx8qxp-adc` driver in the Linux sources, which provides support for the ADC controller of the IMX8QXP and some other SoCs from the i.MX family. The i.MXRT1170 SoC has a similar ADC controller. However, the `imx8qxp-adc` driver does not provide support for all available set of ADC channels. The driver will be updated to support all ADC channels set of the IMXRT1170 SoC. Support for the IMXRT1170 ADC will be enabled via the `DTS` kernel file, using the `nxp,imx8qxp-adc` compatibility string.

New `num_channels` node will be introduced which points out on how many ADC channels are supported by the driver. To support channels `B`, the `channels-b-support` node will be introduced and used.

Raw (unscaled) voltage measurement from channel `Y` of the ADC `X` device can be obtained from the `/sys/bus/iio/devices/iio:deviceX/in_voltageY_raw`.

If support for channel `B` is enabled, then correspondence between the physical ADC channel and the channel number in `sysfs` will define by the following formula:

- for physical channel `X` type `A` corresponding channel in `sysfs` is `X * 2`;
- for physical channel `X` type `B` corresponding channel in `sysfs` is `X * 2 + 1`;

3.2. Effect on Related Products

This project makes the following updates in the related products:

- None

3.3. Changes to User Documentation

This project updates the following user documents:

- None

3.4. Alternative Design

The following alternative design approaches were considered by this project but then discarded for some reason:

- None

4. Test Plan

4.1. Secure Download Area

The downloadable materials developed by this project are available from a secure Web page on the Emcraft Systems web site. Specifically, proceed to the following URL to download the software materials:

for the i.MX RT1170 BSP:

- https://www.emcraft.com/imxrtaddon/imxrt1170-3.0.3/adc

The page is protected as follows:

- Login: *`CONTACT EMCRAFT FOR DETAILS`*
- Password: *`CONTACT EMCRAFT FOR DETAILS`*

4.2. Downloadable Files

The following files are available from the secure download area:

- `linux-adc.patch` - patch to the Linux kernel sources;
- `projects-adc.patch` - patch to the `rootfs` project;
- `rootfs.uImage` - prebuilt bootable Linux image;

4.3. Test Set-Up

4.3.1. Hardware Setup

The following hardware setup is required for the i.MX RT 1170 board:

- The i.MXRT1170 EVK board.

4.3.2. Software Setup

The following software setup is required:

1. Download the files listed in Section: "Downloadable Files" to the top of the Linux i.MX RT installation.
2. Install the BSP, as per the respective "Installing and activating cross development environment" document in the "Software" section on the Emcraft site.
3. From the top of the Linux installation, activate the Linux cross-compile environment by running:

```
$ . ACTIVATE.sh
```

4. From the top of the BSP installation, go to the Linux kernel tree and install the kernel patch, eg:

```
$ cd linux/
$ patch -p1 < ../../linux-adc.patch
```

5. From the top of the Linux installation, go to the `projects` sub-directory, and patch the `rootfs` project:

```
$ cd projects/
$ patch -p1 < ../../projects-adc.patch
```

6. Build the `rootfs` project:

```
$ cd projects/rootfs
$ make
```

4.4. Detailed Test Plan

4.4.1. Test Plan: Demo Project

Perform the following step-wise test procedure:

1. Go to the `projects/rootfs` directory, build the loadable Linux image (`rootfs.uImage`) and copy it to the TFTP directory on the host:

```
$ cd projects/rootfs
$ make
```

2. Boot the loadable Linux image (`rootfs.uImage`) to the target via TFTP and validate that it boots to the Linux shell:

```
=> run netboot
Using ethernet@40424000 device
TFTP from server 192.168.1.96; our IP address is 192.168.1.86
Filename 'imxrt1170/rootfs.uImage'.
Load address: 0x80007fc0
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
...
/ # uname
```

```
Linux
/ #
```

4.4.2. Test Plan: Linux ADC Driver

Perform the following step-wise test procedure:

1. Connect 1.8V to channel 2A of ADC1 (for this, one can wire J56.1 to J26.2)
2. Check that the measurement on channel 2A of ADC1 is near maximum (4096):

```
cat /sys/bus/iio/devices/iio:device0/in_voltage4_raw
4093
```

3. Connect 1.8V to channel 2B of ADC1 (for this, one can wire J56.1 to J26.4)
4. Check that the measurement on channel 2B of ADC1 is near maximum (4096):

```
cat /sys/bus/iio/devices/iio:device0/in_voltage5_raw
4094
```

5. Connect 1.8V to channel 0A of ADC2 (for this, one can wire J56.1 to J9.5)
6. Check that the measurement on channel 0A of ADC2 is near maximum (4096):

```
cat /sys/bus/iio/devices/iio:device1/in voltage0 raw
4093
```

7. Connect 1.8V to channel 0B of ADC2 (for this, one can wire J56.1 to J9.1)
8. Check that the measurement on channel 0B of ADC2 is near maximum (4096):

```
cat /sys/bus/iio/devices/iio:device1/in_voltage1_raw
4091
```

9. Connect GND to channel 2A of ADC1 (for this, one can wire J26.1 to J26.2)
10. Check that the measurement on channel 2A of ADC1 is near minimum (0):

```
/ # cat /sys/bus/iio/devices/iio:device0/in_voltage4_raw
0
```

11. Connect GND to channel 2B of ADC1 (for this, one can wire J26.1 to J26.4)
12. Check that the measurement on channel 2B of ADC1 is near minimum (0):

```
/ # cat /sys/bus/iio/devices/iio:device0/in_voltage5_raw
0
```

13. Connect GND to channel 0A of ADC2 (for this, one can wire J26.1 to J9.5)
14. Check that the measurement on channel 0A of ADC2 is near minimum (0):

```
/ # cat /sys/bus/iio/devices/iio:device1/in voltage0 raw
0
```

15. Connect GND to channel 0B of ADC2 (for this, one can wire J26.1 to J9.1)
16. Check that the measurement on channel 0B of ADC2 is near minimum (0):

```
/ # cat /sys/bus/iio/devices/iio:device1/in_voltage1_raw
0
```

17. Connect a wire with known voltage applied to channel 2A of ADC1.
18. Check that the measurement on channel 2A of ADC1 corresponds to the applied voltage measured with a voltage meter:

```
/ # cat /sys/bus/iio/devices/iio:device0/in_voltage4_raw
3257
/ # cat /sys/bus/iio/devices/iio:device0/in_voltage_scale
0.439453125
/ #
```

(3257 corresponds to 3257 * 0.439453125 = 1.43V)

19. Repeat the tests with channels `2B` of ADC1 (`J26.4`), `0A` of ADC2 (`J9.5`) and `0B` of ADC2 (`J9.1`)