Support SPI in the Linux i.MX RT1050 BSP

Detailed Requirements and Design

rm6916-drad-1_1.doc

RM:	*6916*
Revision:	*1.1*
Date:	*2/11/2024*

TABLE OF CONTENTS

1. Overview

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

This project provides support for the LPSPI controller of the i.MX RT1050 processor in the Linux BSP.

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. Provide support for the i.MX RT1050 LPSPI controller in Linux.

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

3. Provide support for the raw access to the SPI device from user space.

    - *Rationale*: Explicit Customer requirement.
      Implementation: Section: "Design: Raw SPI Device Access".
      Test: Section: "Test Plan: Raw SPI Device Access".

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 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 RT1050 installation.

3.1.2. Design: Linux LPSPI Device Driver

In the i.MXRT1050 SoC, the `LPSPI` controller is compatible with the same controller of some other SoCs from the i.MX family, so the existing `spi-fsl-lpspi.c` driver in the Linux sources, which was initially added to support SPI in the i.MX7ULP SoC, will be used to support the `LPSPI` controller of the i.MXRT1050 SoC.

To enable the driver, the standard `CONFIG_SPI` along with the `CONFIG_SPI_FSL_LPSPI` options will be defined in the kernel `defconfig` file.

In addition to the kernel `defconfig`, the user must configure the `LPSPI` controller in `DTS`. Refer to https://www.kernel.org/doc/Documentation/devicetree/bindings/spi/spi-controller.yaml for details on format of the SPI nodes.

The `DTS` nodes for all 4 instances of the i.MXRT1050 `LPSPI` controller will be predefined in `arch/arm/boot/dts/imxrt1050.dtsi`. The clock driver for the i.MXRT1050 SoC will be updated to support the `LPSPI` clocks. Interrupts and clocks will be defined in the `DTS` nodes according to the processor reference manual. All controllers will be configured to use the DMA. All the SPI nodes in the `.dtsi` file will be disabled by default.

Final tuning of the kernel run-time configuration, such as defining the `pinctrl` settings for a custom connection of an SPI device to the `LPSPI` controller, defining the `chip-select` signal, customizing the clocks etc, will be done in the user `DTS` file.

This project will provide an example for configuring the `LPSPI1` controller. There are no SPI devices on the IMXRT1050-EVK board connected to the LPSPI controller, so external SPI Flash connected via the SD TF to TF Flexible Card Extension cable will be used to demonstrate reading the SPI device registers.

The `lpspi1` node will be configured in the `rootfs.dts.IMXRT105X_NXPEVK` file as follows:

```
&lpspi1 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl lpspi1>;
        status = "okay";
        cs-gpios = <&gpio3 13 GPIO ACTIVE LOW>;
        num-chipselects = <1>;
...
};
&iomuxc {
...
        pinctrl lpspi1: lpspi1grp {
               fsl,pins = <
                        MXRT1050_IOMUXC_GPIO_SD_B0_00_LPSPI1_SCK          MXRT10XX_PAD_CFG_SPI
                        MXRT1050 IOMUXC GPIO SD B0 02 LPSPI1 SDO          MXRT10XX PAD CFG SPI
                        MXRT1050 IOMUXC GPIO SD B0 03 LPSPI1 SDI          MXRT10XX PAD CFG SPI
                        /* CS#1 (SPI Flash) */
                        MXRT1050 IOMUXC GPIO SD B0 01 GPIO3 IO13
MXRT10XX_PAD_CFG_GPIO
                 >;
       };
};
```

This example assigns the i.MXRT1050 `pinctrl` PADs `GPIO_SD_B0_00`, `GPIO_SD_B0_01`, `GPIO_SD_B0_02` and `GPIO_SD_B0_03` to the `LPSPI1` controller and selects the `LPSPI1_SCK`, `GPIO`, `LPSPI1_SDO` and `LPSPI1_SDI` functions for these PADs (refer to the board schematics and to the processor reference manual for details on possible `IOMUXC` configurations).

The single `chip-select` is configured to the `GPIO3 13` signal in the example.

Due to the fact that the `GPIO_SD_B0_00`, `GPIO_SD_B0_01`, `GPIO_SD_B0_02` and `GPIO_SD_B0_03` PADs are used for the SD card controller, the changes for `LPSPI1` will be provided under the `#if defined(SPI_FLASH_ON_LPSPI1)` condition. If the `SPI_FLASH_ON_LPSPI1` is defined in `DTS`, the SD-card controller will be disabled.

3.1.3. Design: Raw SPI Device Access

Linux provides a special `spidev` device driver to allow raw accesses to SPI devices from the user space: https://www.kernel.org/doc/Documentation/spi/spidev.

The standard Linux `CONFIG_SPI_SPIDEV` option will be defined in the kernel `defconfig` file to enable the `spidev` driver. The universal compatibility string "linux,spidev" will be added to the `spidev` driver so that it can be used in the project `DTS`.

The following changes will be made to `rootfs.dts.IMXRT105X_NXPEVK` to link an external SPI Flash to the `spidev` Linux device:

```
&lpspi1 {
...
#if defined(LPSPI USE SPIDEV)
       spidev: spidev@0 {
               status = "okay";
               compatible = "linux,spidev";
               spi-max-frequency = <33000000>;
               reg = <0>;
       };
...
#endif
};
```

The following test program `spidev_flash` will be included to the project root file system as an example on how to read the Flash ID of an SPI Flash device from the user-space applications:

```
/*
 * Sample application that makes use of the SPIDEV interface
 * to access an SPI slave device. Specifically, this sample
 * reads a Device ID of a JEDEC-compliant SPI Flash device.
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>

int main(int argc, char **argv)
{
        char *name;
        int fd;
        struct spi ioc transfer xfer[2];
        unsigned char buf[32], *bp;
        int len, status;

        name = argv[1];
        fd = open(name, O RDWR);
        if (fd < 0) {
                perror("open");
                return 1;
        }

        memset(xfer, 0, sizeof xfer);
        memset(buf, 0, sizeof buf);
        len = sizeof buf;
```

```
        /*
         * Send a GetID command
         */
        buf[0] = 0x9f;
        len = 6;
        xfer[0].tx_buf = (unsigned long)buf;
        xfer[0].len = 1;

        xfer[1].rx_buf = (unsigned long) buf;
        xfer[1].len = 6;

        status = ioctl(fd, SPI_IOC_MESSAGE(2), xfer);
        if (status < 0) {
                perror("SPI_IOC_MESSAGE");
                return -1;
        }

        printf("response(%d): ", status);
        for (bp = buf; len; len--)
                printf("%02x ", *bp++);
        printf("\n");

        return 0;
}
```

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 RT1050 BSP:

https://www.emcraft.com/imxrtaddon/imxrt1050/lpspi/

The page is protected as follows:

- Login: *CONSTACT EMCRFT FOR DETAILS*

- Password: *CONSTACT EMCRFT FOR DETAILS*

4.2. Downloadable Files

The following files are available from the secure download area:

- `linux-lpspi.patch` - patch to the Linux kernel sources;
- `projects-lpspi.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 1050 board:

- The i.MXRT1050 EVK board.
- Micron M25P32 SPI Flash Memory, as a separate chip.
- Connect the SPI Flash memory via the "SD TF to TF Flexible Card Extension cable Extender Adapter" attached to the TF Card Slot `J20`. Details of those connection are as follows:

SPI Memory Signal/Function	***J20 Pin#***	***MCU Signal***	***MCU PAD***
CS#/Chip Select	5	LPSPI1_PCS0	GPIO_SD_B0_01
SCLK/Clock Input	3	LPSPI1_SCK	GPIO_SD_B0_00
SI/Serial Data Input	7	LPSPI1_SDO	GPIO_SD_B0_02
SO/Serial Data Output	8	LPSPI1_SDI	GPIO_SD_B0_03

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-lpspi.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-lpspi.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 'imxrt1050/rootfs.uImage'.
Load address: 0x80007fc0
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
...
/ # uname
Linux
/ #
```

4.4.2. Test Plan: Linux LPSPI Driver

Perform the following step-wise test procedure:

1. From the Linux shell, run `dmesg` and verify that there are no error messages related to SPI or LPSPI:

```
/ # dmesg
...
/ # dmesg | grep spi
/ #
```

2. Verify that the `spi0` master is registered in the `sysfs`:

```
/ # ls /sys/class/spi master/
spi0
/ #
```

4.4.3. Test Plan: Raw SPI Device Access

1. Replace the SD card in the SD card slot by the SPI Flash memory `M25P32` using "SD TF to TF Flexible Card Extension cable Extender Adapter".
2. Make sure the `spidev` device exists:

```
/ # ls -l /dev/spidev0.0
crw-------    1 root     root      153,   0 Jan  1 00:00 /dev/spidev0.0
/ #
```

3. Make sure the `spidev_flash` application reads the correct Flash ID over SPI (first 3 bytes must be `0x20 0x20 0x16` for the `M25P32` Flash connected the IMXRT1050-EVK board):

```
/ # spidev_flash /dev/spidev0.0
response(7): 20 20 16 10 00 00
```

```
/ #
```