Develop Linux Driver for PCAL6524

Detailed Requirements and Design

rt122285-drad-1_1.doc

RT#:	*122285*
Revision:	*1.1*
Date:	*28/6/2016*

TABLE OF CONTENTS

1. OVERVIEW 3

2. REQUIREMENTS 3

2.1. Detailed Requirements 3
2.2. Detailed Non-Requirements 3

3. DESIGN 3

3.1. Detailed Design 3
3.1.1. Design: Demo Project 3
3.1.2. Design: Device Driver 4
3.1.3. Design: Loadable Image 4
3.2. Changes to User Interfaces 4
3.3. Changes to Internal APIs 4
3.4. Effect on Related Products 4
3.5. Changes to User Documentation 4
3.6. Alternative Design 4

4. TEST PLAN 5

4.1. Secure Download Area 5
4.2. Downloadable Files 5
4.3. UM10868 Board 5
4.4. Test Set-Up 5
4.5. Detailed Test Plan 6
4.5.1. Test Plan: Demo Project 6
4.5.2. Test Plan: Device Driver 6
4.5.3. Test Plan: Bootable Image 7

1. Overview

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

Develop a Linux device driver for the PCAL6524 I2C GPIO device.

2. Requirements

2.1. Detailed Requirements

The following are the requirements for this project:

1. Provide a demo project (Linux configuration) combining all the requirements in this project.
    - *Rationale*: To provide a reference software baseline for the Customer.
      Implementation: Section: "Design: Demo Project".
      Test: Section: "Test Plan: Demo Project".

2. Provide a Linux device driver for the PCAL6524 I2C GPIO device. As the functional acceptance test, use the standard Linux GPIOLIB framework to control and monitor the GPIO signals of the PCAL6524 device.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Device Driver".
      Test: Section: "Test Plan: Device Driver".

3. Provide a bootable Linux image for the above embedded project. Instructions for loading the image must be provided.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Loadable Image".
      Test: Section: "Test Plan: Bootable Image".

2.2. Detailed Non-Requirements

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

1. No support for I2C devices other than the PCAL6524 I2C GPIO device.
    - *Rationale*: Explicit non-requirement.

3. Design

3.1. Detailed Design

3.1.1. Design: Demo Project

This project will develop a demo Linux configuration ("embedded project") that will provide all the required software functionality from a single bootable Linux image. The project will be called `pcal6524` (and will reside in a `projects/pcal6524` directory, relative to the top of the STM32F4 SOM Linux installation).

Customer will be able to install the project source tree to the development host machine, build the project, install it to the Emcraft STM32F4 SOM board and validate all requirements from the loaded Linux image.

3.1.2. Design: Device Driver

Kernel support for the PCAL6524 I2C GPIO device implies changes at several separate levels:

1. A Linux device driver will be implemented in the kernel source tree as `linux/drivers/gpio/pcal6524.c`. The Linux (uClinux) BSP for the Emcraft STM32F4 SOM Starter Kit, used as the hardware reference platform in this project, makes use of the Linux kernel v.4.2.
2. `Makefile` and `Kconfig` in `linux/drivers/gpio/` will be extended to build the new device driver whenever the corresponding kernel configuration option is enabled or not (refer to Section: "Changes to User Interfaces").
3. The `projects/pcal6524/pcal6524.dts.STM32F4X9` device tree file will describe a platform-specific configuration for the PCAL6524 chip connected to the `I2C_1` bus.

3.1.3. Design: Loadable Image

The customer will be provided with a loadable Linux image `pcal6524.uImage` for the demo project described in Section: "Design: Demo Project".

The loadable Linux image is just a convenience. The customer will be able to build exactly the same image from the demo project sources.

3.2. Changes to User Interfaces

This project makes the following changes to the existing user interfaces:

- A new kernel configuration option (`CONFIG_GPIO_PCAL6524`) will be added to `linux/device/gpio/Kconfig`. This option will enable the Linux device driver for the PCAL6524 I2C GPIO device in the kernel configuration.

3.3. Changes to Internal APIs

This project makes the following important changes to the existing internal APIs:

- None.

3.4. Effect on Related Products

This project makes the following updates in the related products:

- None. That said, the new device driver could be easily ported to the other Linux platforms.

3.5. Changes to User Documentation

This project updates the following user documents:

- None.

3.6. 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:

- https://www.emcraft.com/nxp

The page is protected as follows:

- Login: `nxp`
- Password: `chee21a7`

4.2. Downloadable Files

The following files are available from the secure download area in the `rt122285` folder for this release:

- `rt122285-linux.patch` - Patch to the Linux kernel sources developed by this project;
- `pcal6524.tgz` - The Linux configuration (embedded project) specifically developed to demonstrate the requirements of this project.
- `pcal6524.uImage` - The bootable Linux image for the above embedded project.

Refer to the below sections for the instructions on how to install and use these files.

4.3. UM10868 Board

This project makes use of the NXP UM10868 board, which is PCAL6524 demonstration board. The board is hand-wired to the `I2C_1` interface available on the breadboard area of the SOM-BSB-EXT development board, included with the STM32F4 SOM Starter Kit, as follows:

UM10868	**Emcraft STM32F4 SOM**
CN1 pin 2 (GND)	P9 pin 4 (GND)
CN1 pin 4 (5V_TSTR)	P9 pin 1 (3.3V)
CN1 pin 3 (SCL)	P9 pin 8 (I2C_1-SCL)
CN1 pin 5 (SDA)	P9 pin 9 (I2C_1-SDA)

Additional comments:

- The I2C address of the PCAL6524 GPIO device is hardwired to 0x23 by closing pins 1 and 2 of J10 via a jumper.
- Refer to the following page on the Emcraft web site for the hardware materials describing the STM32F4 SOM Starter Kit and SOM-BSB-EXT development board:
    - http://www.emcraft.com/products/224#hardware

4.4. Test Set-Up

The following set-up is required for execution of the test plan in this project:

- Emcraft STM32F4 SOM Starter Kit. Refer to the following page for details:

  - http://www.emcraft.com/products/224#starter-kit
- The U-Boot binary (version Release 2.0.0) installed to the Emcraft STM32F4 SOM board. All kits purchased from Emcraft starting from June 2016 come with this U-boot pre-installed to the SOM at the factory.
- The NXP UM10868 board connected to the STM32F4 SOM Starter Kit as described in Section: "UM10868 Board".

4.5. Detailed Test Plan

4.5.1. Test Plan: Demo Project

The following step-wise test procedure will be used:

1. From the top of the STM32F4 SOM Linux installation (Release 2.0.0), activate the Linux cross-compile environment by running:

```
. ACTIVATE.sh
```

2. Go to the Linux kernel tree:

```
cd linux/
```

3. Install the patch developed by this project:

```
patch -p1 < rt122285-linux.patch
...
```

4. From the top of the STM32F4 SOM Linux installation, go to the `projects` sub-directory:

```
cd projects
```

5. Unpack the `pcal6524` project developed by this project:

```
tar xvf pcal6524.tgz
```

6. Go to the newly installed `pcal6524/` directory and build the loadable Linux image (`pcal6524.uImage`):

```
cd pcal6524
make
```

7. Load the built Linux image (`pcal6524.uImage`) to the target as described in the following application note:
   - http://www.emcraft.com/som/stm32f4-200/loading-linux-images-via-ethernet-and-tftp

4.5.2. Test Plan: Device Driver

The following step-wise test procedure will be used:

1. Reset the target and let Linux image installed to the Flash come up to the shell.
2. Scan the `I2C_1` bus and make sure that the PCAL6524 GPIO device is found at 0x23. The `UU` indicates that the device is there but taken by a device driver (which is expected):

```
/ # i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- UU -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
/ #
```

3. Go the GPIOLIB directory in the `sysfs` file system to reduce the pathnames in the follow up test commands:

```
~ # cd /sys/class/gpio/
```

4. Validate that the PCAL6524 chip is available at position 488:

```
/sys/class/gpio # cat gpiochip488/label
pcal6524
/sys/class/gpio #
```

5. Configure the GPIO 20 (488 + 20 = 508) as an output and set its value to zero:

```
/sys/class/gpio # echo 508 > export
/sys/class/gpio # echo out > gpio508/direction
/sys/class/gpio # echo 0 > gpio508/value
```

6. Validate a zero voltage on pin P0_4 of CN5 on the UM10868 board with a voltmeter.
7. Turn on the GPIO 20 (488 + 20 = 508) and validate a 3.3V voltage on pin P0_4 of CN5 on the UM10868 board with a voltmeter.

```
/sys/class/gpio # echo 1 > gpio508/value
```

8. Configure the GPIO 20 as an input:

```
/sys/class/gpio # echo in > gpio508/direction
```

9. Connect the GPIO 20 (CN5.P0_4) to the ground (pin 2 of J5), validate that its value is zero:

```
/sys/class/gpio # cat gpio508/value
0
/sys/class/gpio #
```

10. Connect the GPIO 20 (CN5.P0_4) to the 3.3V (pin 1 of J5), validate that its value is 1:

```
/sys/class/gpio # cat gpio508/value
1
/sys/class/gpio #
```

4.5.3. Test Plan: Bootable Image

The following step-wise test procedure will be used:

1. Install the pre-built Linux image (`pcal6524.uImage`) to the target as described in the following application note:
   - o http://www.emcraft.com/som/stm32f4-200/installing-linux-images-to-flash
2. Repeat the functional test procedure documented in Section: "Test Plan: Device Driver".