PMIC hardware components

Contents

Stable: 20.11.2020 - 17:11 / Revision: 20.11.2020 - 17:11

Stable: 20.11.2020 - 17.11 / Revision: 20.11.2020 - 17.11

Contents

1 Article purpose

The purpose of this article is to:

- list the PMIC hardware components that might be integrated in the different boards
- link these components to the corresponding software framework(s)
- point to the datasheet(s) of these components
- explain, when necessary, how to configure these components.

2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
PMIC	STPMIC1	OP-TEE overview	Regulator overview Watchdog overview U-Boot	

3 STPMIC1

3.1 Description

The STPMIC1 is a device that handles the power supplies for some STM32MP boards. The STPMIC1 main features are:

- configuration via I²C bus,
- 10 regulators of different kinds with over-current protection,
- 3 power switches for USB supplies,
- a power-on key input,
- a watchdog,
- a thermal protection,
- a NORMAL mode and an ALTERNATE mode that is used for system suspend.

The user manual is available here: NOT YET AVAILABLE

3.2 Support in Linux Kernel

The STPMIC1 driver is used to handle STPMIC1 Power Management Integrated Circuit (PMIC). The driver is composed of several parts:

- The core driver
- The regulator driver
- The watchdog driver
- The input driver

The thermal feature is not supported.

3.2.1 Core driver

The core implements the Multi-Function Devices Framework API.

- It is probed by the I²C framework.
- It handles communication with the PMIC via I²C.
- It probes the others STPMIC1 drivers (input, watchdog....).
- It acts as an interrupt controller for the other drivers.

Source code:

```
include/linux/mfd/stpmic1.h
drivers/mfd/stpmic1.c
```

Bindings are described in:

```
Documentation/devicetree/bindings/mfd/st,stpmic1.txt
```

3.2.2 Regulator driver

The driver implements the Regulator framework API

- Each BUCK, LDO or POWER SWITCH inside the STPMIC1 device is presented as a voltage regulator.
- Power switches are presented as fixed voltage sources. Voltage can not be set.
- The driver does not handle the suspend configuration. This is done by the Secure Monitor.

Source code:

```
drivers/regulator/stpmic1_regulator.c
```

Bindings are described in:

```
Documentation/devicetree/bindings/regulator/st,stpmic1-regulator.txt
```

3.2.3 Watchdog driver

The driver implements the Watchdog framework API

When enabled, a watchdog device is available for the user-land. As soon as a user has started to write in the watchdog it is armed in the PMIC. When the watchdog timer expires, the PMIC shuts down.

Source code:

```
drivers/watchdog/stpmic1_wdt.c
```

Bindings are described in:

```
Documentation/devicetree/bindings/watchdog/st,stpmic1-wdt.txt
```

3.2.4 Input driver

The driver implements the Input framework API.

The driver exposes a single key (KEY_POWER) that can be used as any standard input device in /dev/input/eventX

Source code:

```
drivers/input/misc/stpmic1_onkey.c
```

Bindings are described in:

```
Documentation/devicetree/bindings/input/st,stpmic1-onkey.txt
```

3.2.5 Kernel Configuration

With kernel menuconfig, enable following configurations:

- core part: MFD_STPMIC1
- regulator: REGULATOR_STPMIC1

- watchdog: STPMIC1_WATCHDOG
- input: INPUT_STPMIC1_ONKEY

3.3 Support in U-BOOT

STPMIC1 is used by U-Boot to:

- Control regulators used by other drivers (mmc-supply for SDCard for example, usb vbus)
- Trace pmic startup reason

STMPIC is supported with existing uclass of the 'Driver Model' described in doc/driver-model/pmic-framework.rst .

- pmic :
  - uclass: drivers/power/regulator/regulator-uclass.c , CONFIG_DM_PMIC
  - driver: drivers/power/pmic/stpmic1.c , CONFIG_PMIC_STPMIC1
- regulator:
  - uclass: drivers/power/pmic/pmic-uclass.c , CONFIG_DM_REGULATOR
  - driver: drivers/power/regulator/stpmic1.c , CONFIG_DM_REGULATOR_STPMIC1
- sysreset:
  - uclass: drivers/sysreset/sysreset-uclass.c , CONFIG_SYSRESET
  - code in pmic driver

The STPMIC1 is available with the existing commands:

- pmic (CONFIG_CMD_PMIC)
- regulator (CONFIG_CMD_REGULATOR)
- poweroff (CONFIG_CMD_POWEROFF)

SPTMIC1 driver supports configuration via device-tree; the bindings, same as kernel, are described in:

```
Documentation/devicetree/bindings/regulator/st,stpmic1-regulator.txt
```

3.4 Support in Cortex-A7 Secure

3.4.1 TF-A

STPMIC1 is used by TF-A firmware to:

- Configure DDR power supplies.
- Configure the regulators for system suspend and system shutdown.

Driver source code:

```
include/drivers/st/stpmic1.h
drivers/st/pmic/stpmic1.c
```

SPTMIC1 driver supports configuration via device-tree, bindings are described in:

```
docs/devicetree/bindings/power/st,stpmic1.txt
```

An example of PMIC configuration with device tree can be found in this file, node &i2c4:

```
fdts/stm32mp15xx-edx.dtsi
```

When using a PMIC on a board, TF-A is configured to use a secure I²C (I2C4 or I2C6). By default on STMicroelectronics boards, I2C4 is used.

Using another I²C (non-secure) is not supported in the TF-A implementation made by STMicroelectronics. But it is possible to modify the code and device tree files to support it.

Here is an example with I2C2:

- Copy the chosen I²C node from Linux device tree in the TF-A device tree file for your board, as well as its dependencies (like pinctrl). You can also remove the properties not used by TF-A as done in the example below

```
/ {
...
        soc {
        ...
 +                i2c2: i2c@40013000 {
 +                        compatible = "st,stm32mp15-i2c";
 +                        reg = <0x40013000 0x400>;
 +                        clocks = <&rcc I2C2_K>;
 +                        resets = <&rcc I2C2_R>;
 +                        #address-cells = <1>;
 +                        #size-cells = <0>;
 +                        status = "disabled";
 +                };
```

- change pmic node parent to this newly added I²C

```
& i2c2 {
        ...
        pmic: stpmic@33 {   # You may also require to change this I²C address (33)
        ...
};
```

- remove in PMIC node the property:

```
secure-status = "okay";
```

- Add this I²C base address in plat/st/stm32mp1/stm32mp1_def.h :

```
#define I2C2_BASE                       U(0x40013000)
```

- Add a case for this I²C in register_periph_iomem() function, in the file plat/st/stm32mp1/stm32mp1_shared_resources.c . It should be placed with the other non-secure peripherals (UARTs and IWDG2):

```
        case UART7_BASE:
        case UART8_BASE:
        case IWDG2_BASE:
+        case I2C2_BASE:
                /* Allow drivers to register some non secure resources */
                VERBOSE("IO for non secure resource 0x%x\n",
                        (unsigned int)base);
```

- Add the clock of this I²C in the stm32mp1_clk_gate[] table in the file drivers/st/clk/stm32mp1_clk.c

```
+        _CLK_SC_SELEC(RCC_MP_APB1ENSETR, 22, I2C2_K, _I2C12_SEL),
```

- Regarding the GPIO ports used by I2C2: in this stm32mp1_clk_gate[] table, remove the #ifdef IMAGE_BL2 around GPIO banks entries if present

3.4.2 OP-TEE

STPMIC1 is used by OP-TEE OS to configure the regulators for system suspend and system shutdown.

Driver source code:

```
core/include/drivers/stpmic1.h
core/drivers/stpmic1.c
```

Power Management Integrated Circuit

Open Portable Trusted Execution Environment

Application programming interface

Low-dropout regulator

Multifunction device

Trusted Firmware for Arm Cortex-A

Doubledata rate (memory domain)

input/output

Reset and Clock Control

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Boot Loader stage 2

Operating System

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

Stable: 11.06.2020 - 12:50 / Revision: 11.06.2020 - 12:12

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

Stable: 17.02.2021 - 19:40 / Revision: 16.02.2021 - 16:25

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

Stable: 01.03.2021 - 10:54 / Revision: 01.03.2021 - 10:53

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

Stable: 31.01.2020 - 13:22 / Revision: 31.01.2020 - 13:13

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer