GPIO device tree configuration

Contents

Stable: 12.08.2021 - 10.14 | Revision: 12.08.2021 - 10.15

A quality version of this page, approved on *12 August 2021*, was based off this revision.

Contents

1 Purpose 4
2 DT bindings documentation 5
3 DT configuration 6
3.1 DT configuration (STM32 level) 6
3.2 DT configuration (board level) 6
3.3 DT configuration examples 6
3.3.1 How to add a GPIO dedicated to a device connected on the board 6
4 How to configure GPIOs using STM32CubeMX 8
5 References 9

1 Purpose

The purpose of this article is to explain how to configure the GPIO internal peripheral through the **GPIOLib framework when this peripheral is assigned to Linux®OS**. The configuration is performed using the device tree[1].

To better understand I/O management, it is recommended to read the I/O pins overview article [2].

This article also provides an example explaining how to add a new GPIO in the device tree.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

The STM32 GPIO bindings are composed of:

- Bindings for GPIO controller [3]
- Bindings for devices using a GPIO [4]

It is recommended to read these reference documents if you have never played with GPIO DT.

3 DT configuration

3.1 DT configuration (STM32 level)

The GPIO controller node (gpio controller) is located in the pin controller node that is declared Inside the SoC dtsi file, *stm32mp151.dtsi*[5]. See Device tree for more explanations about device tree files split.

There is one gpio controller node per GPIO bank.

Refer to Pin controller configuration and to GPIO controller node bindings[3] for explanations on the gpio controller node.

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

Generic guidelines for adding a GPIO to a client device can be found in the document "GPIO bindings for board" [4].

- Below an example of basic GPIO usage extracted from "GPIO bindings for board" [4]:

GPIO mappings are defined in the ***consumer device node***, through a property named *<function>-gpios*, where <function> is requested by the Linux driver through gpiod_get().

```
foo_device {
                ...
                led-gpios = <&gpioa 15 GPIO_ACTIVE_HIGH>, /* red */
                            <&gpioa 16 GPIO_ACTIVE_HIGH>, /* green */
                            <&gpioa 17 GPIO_ACTIVE_HIGH>; /* blue */

                power-gpios = <&gpiob 1 GPIO_ACTIVE_LOW>;
        };
```

Where:

- *&gpiox*: phandle on gpio-controller node.
- *15*: line offset in gpio-controller.
- *GPIO_ACTIVE_HIGH*: flag to be used for the GPIO. The list of all available GPIO flags can be found in Linux kernel source code[6].

3.3 DT configuration examples

3.3.1 How to add a GPIO dedicated to a device connected on the board

The example below shows how to add a GPIOB5 (PB5 on schematics) to a foo_device.

```
foo_device {
                ...
                x-gpios = <&gpiob 5 (GPIO_ACTIVE_HIGH | GPIO_PULL_UP)>; /* x function */
                ...
        };
```

Note: GPIO_ACTIVE_HIGH means that the GPIO line is driven by the logical level 1 (while GPIO_ACTIVE_LOW means that it is driven by the logical level 0). GPIO_PULL_UP means that the internal pull-up resistor is enabled.

4 How to configure GPIOs using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- Device tree, Device tree mechanism
- Overview of GPIO pins, Overview of GPIO pins article
- 3.03.1 Kernel device tree bindings documentation - gpio.txt, GPIO device tree bindings
- 4.04.14.2 GPIO bindings for board
- stm32mp151.dtsi , STM32MP151 device tree file
- include/dt-bindings/gpio/gpio.h

}}

Linux® is a registered trademark of Linus Torvalds.

Operating System

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.)

Device Tree

uniprocessor