TIM device tree configuration

Stable: 09.10.2019 - 14:46 / Revision: 07.10.2019 - 11:47

Contents

1 Article purpose

The purpose of this article is to explain how to configure the *timer (TIM)*[1] **when the peripheral is assigned to Linux® OS**:

- Configuring the timer **peripheral** to enable PWM, trigger or quadrature encoder.
- Configuring the **board**, e.g. TIM pins.

The configuration is performed using the **device tree mechanism**[2].

It is used by the TIM Linux driver that registers relevant information in PWM and IIO frameworks.

2 DT bindings documentation

The *TIM internal peripheral*[1] is a multifunction device (MFD).

Each function is represented by a separate DT binding document:

- *STM32 TIM MFD device tree bindings*[3] document deals with core resources (e.g. registers, clock, DMAs)
- *STM32 TIM PWM device tree bindings*[4] document deals with PWM resources (e.g. PWM input/output pins)
- *STM32 TIM IIO trigger/encoder device tree bindings*[5] document deals with other internal peripheral triggering and quadrature encoder resources

3 DT configuration

This hardware description is a combination of both STM32 microprocessor and board device tree files. Refer to Device tree for more explanations about device tree file split.

The **STM32CubeMX** can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

TIM nodes are declared in stm32mp157c.dtsi[6].

DT root node (e.g. timers1...) and **DT child nodes** describe the TIM features such as:

- PWM
- trigger and quadrature encoder

They also describe hardware parameters such as registers address, clock and DMA.

```
timers1: timer@address {
        compatible = "st,stm32-timers";                    /* timer common resources */
        ...
        pwm {
                compatible = "st,stm32-pwm";               /* PWM*/
        };
        timer@0 {
                compatible = "st,stm32h7-timer-trigger";   /* trigger and quadrature enc
                reg = <0>;                                 /* trigger identifier (e.g. 0
        };
};
```

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

3.2 DT configuration (board level)

This part is used to configure and enable the TIM hardware used on the board:

- Enabling **DT root node** for the TIM instances in use (e.g timers1...) by setting **status = "okay";**
- Enabling **DT child node(s)** for the feature(s) in use (PWM input/output, trigger and quadrature encoder) by setting **status = "okay";**
- Configuring pins in use via pinctrl through **pinctrl-0**, **pinctrl-1** and **pinctrl-names**.

3.3 DT configuration examples

3.3.1 TIM configured in PWM mode and trigger source

The example below shows how to configure **TIM1 channel 1** to act as:

- **PWM output on PE9**, e.g. TIM1_CH1 (See pinctrl device tree configuration and GPIO internal peripheral)
- **trigger source** (synchronous with PWM) for other internal peripheral such as STM32 ADC

Note: The PWM output does not require any DMA channel. Disable them if they are configured by default in the .dtsi file.

```
pwm1_pins_a: pwm1-0 {
        pins {
                pinmux = <STM32_PINMUX('E', 9, AF1)>;        /* select TIM1_CH1 alternate
                bias-pull-down;
                drive-push-pull;
                slew-rate = <0>;
        };
};

pwm1_sleep_pins_a: pwm1-sleep-0 {
        pins {
                pinmux = <STM32_PINMUX('E', 9, ANALOG)>;     /* configure 'PE9' as analog
        };
};
```

```
&timers1 {
        status = "okay";
        /delete-property/dmas;                               /* spare all DMA channels sin
        /delete-property/dma-names;
        pwm {
                pinctrl-0 = <&pwm1_pins_a>;                  /* configure PWM on TIM1_CH1
                pinctrl-1 = <&pwm1_sleep_pins_a>;
                pinctrl-names = "default", "sleep";
                status = "okay";                             /* enable PWM on TIM1 */
        };
        timer@0 {
                status = "okay";                             /* enable trigger on TIM1 */
        };
};
```

3.3.2 TIM configured in PWM input capture mode

The example below shows how to configure **TIM1 channel 1** in PWM input capture mode (e.g. period and duty cycle):

- Configure **PWM input on PE9**, e.g. TIM1_CH1 (See pinctrl device tree configuration and GPIO internal peripheral)

A DMA channel is required and must be configured depending on the PWM input channel:

- Select **DMA channel 1**, "ch1", to capture **PWM input channel 1 and/or 2**
- Select **DMA channel 3**, "ch3", to capture **PWM input channel 3 and/or 4**

```
pwm1_in_pins_a: pwm1-in-0 {
        pins {
                pinmux = <STM32_PINMUX('E', 9, AF1)>;        /* select TIM1_CH1 alternate
                bias-disable;
        };
};

pwm1_in_sleep_pins_a: pwm1-in-sleep-0 {
        pins {
                pinmux = <STM32_PINMUX('E', 9, ANALOG)>;     /* configure 'PE9' as analog
        };
};
```

```
&timers1 {
        status = "okay";
        dmas = <&dmamux1 11 0x400 0x5>;                      /* Enable DMA "ch1" for PWM i
        dma-names = "ch1";
        pwm {
                pinctrl-0 = <&pwm1_in_pins_a>;               /* configure PWM input pins,
                pinctrl-1 = <&pwm1_in_sleep_pins_a>;
                pinctrl-names = "default", "sleep";
                status = "okay";                             /* enable PWM on TIM1 */
        };
};
```

3.3.3 TIM configured as quadrature encoder interface

The example below shows how to configure **TIM1** to interface with a quadrature encoder:

- **Configure PE9 and PJ11 as encoder input pins**, e.g. TIM1_CH1, TIM1_CH2 (see pinctrl device tree configuration and GPIO internal peripheral)

```
tim1_in_pins_a: tim1-in-pins-0 {
        pins {
                pinmux = <STM32_PINMUX('E', 9, AF1)>, /* TIM1_CH1 */
                         <STM32_PINMUX('J', 11, AF1)>; /* TIM1_CH2 */
                bias-disable;
        };
};

tim1_in_pins_sleep_a: tim1-in-pins-sleep-0 {
        pins {
                pinmux = <STM32_PINMUX('E', 9, ANALOG)>, /* TIM1_CH1 */
                         <STM32_PINMUX('J', 11, ANALOG)>; /* TIM1_CH2 */
        };
};
```

```
&timers1 {
        status = "okay";
        /delete-property/dmas;                               /* spare all DMA channels sin
        /delete-property/dma-names;
        timer@0 {
```

```
                pinctrl-0 = <&tim1_in_pins_a>;                  /* configure TIM1_CH1 and TIM
                pinctrl-1 = <&tim1_in_pins_sleep_a>;
                pinctrl-names = "default", "sleep";
                status = "okay";                                /* enable Encoder interface m
        };
};
```

4 How to configure the DT 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:

1. ↑ [1.01.1] TIM internal peripheral
2. ↑ Device tree
3. ↑ Documentation/devicetree/bindings/mfd/stm32-timers.txt , STM32 TIM MFD device tree bindings
4. ↑ Documentation/devicetree/bindings/pwm/pwm-stm32.txt , STM32 TIM PWM device tree bindings
5. ↑ Documentation/devicetree/bindings/iio/timer/stm32-timer-trigger.txt , STM32 TIM trigger/encoder device tree bindings
6. ↑ stm32mp157c.dtsi , STM32.dtsi file

Operating System

Pulse Width Modulation

Device Tree

Multifunction device

Industrial I/O Linux subsystem

Direct Memory Access