USBPHYC device tree configuration

Stable: 12.02.2019 - 09:18 / Revision: 21.01.2019 - 15:01

Contents

1 Article purpose

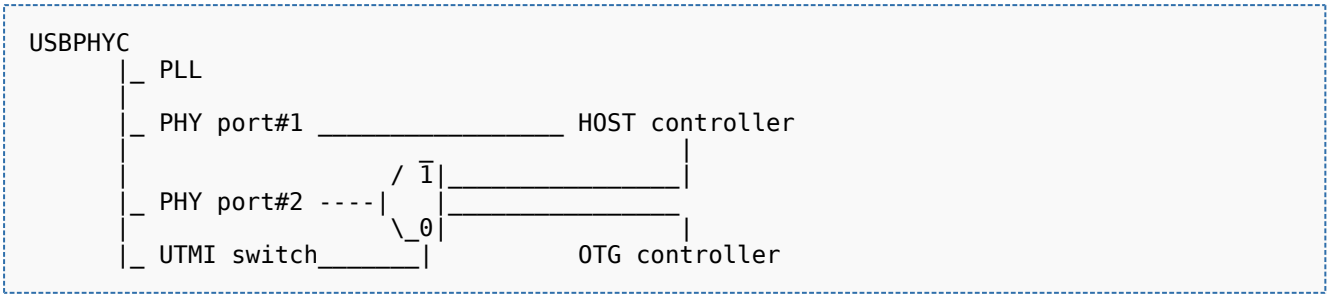
This article explains how to configure the **USBPHYC** internal peripheral when it is assigned to the Linux® OS. In that case, it is controlled by the PHY framework.

The configuration is performed using the device tree mechanism.

It is used by the *USBPHYC Linux driver*[1] which registers the relevant information in PHY framework.

2 DT bindings documentation

USBPHYC device tree bindings[2] describe all the required and optional functions.



3 DT configuration

This hardware description is a combination of the **STM32 microprocessor** device tree files (*.dtsi* extension) and **board** device tree files (*.dts* extension). See the Device tree for an explanation of the device tree file split.

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)

The USBPHYC node is declared in stm32mp157c.dtsi[3].

- root node e.g. **usbphyc** describes the USBPHYC hardware block parameters such as registers, clocks, resets and supplies.
- child nodes e.g. **usbphyc_port0** and **usbphyc_port1** describe the two high speed PHY ports: *port#1* and *port#2*.

```
usbphyc: usbphyc@address {
        compatible = "st,stm32mp1-usbphyc";
        ...                                          /* usbphyc resources: registers, clock
        usbphyc_port0: usb-phy@0 {
                ...                                  /* usbphyc HS PHY port#1 */
        };
        usbphyc_port1: usb-phy@1 {
                ...                                  /* usbphyc HS PHY port#2 */
        };
};
```

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)

Follow the sequences described in the below chapters to configure and enable the USBPHYC on your board.

The '**usbphyc**' **root node** must be filled in:

- Enable the USBPHYC block by setting **status = "okay".**
- Configure the USBPHYC 3V3 regulator[4] by setting **vdd3v3-supply = <&your_regulator>**.

USB HS PHY ports require an external 3V3 power supply to be provided at VDD3V3_USBHS pin.

The **child nodes** for each port may be tuned:

- Optional: create a **usb_phy_tuning** node that can take optional parameters in DT root folder ('/')
- Optional: add '**st,phy-tuning = <&usb_phy_tuning>**' in '**usbphyc_port0**' and/or '**usbphyc_port1**' node to use this tuning.

It may be necessary to adjust the phy settings to compensate parasitics, which can be due to USB connector/receptacle, routing, ESD protection component.

Optional tuning parameter list is available in *USBPHYC device tree bindings*[2].

3.3 DT configuration example

The example below shows how to enable and configure USBPHYC ports in the board file

```
&usbphyc {
        vdd3v3-supply = <&vdd_usb>;                     /* references the 3V3 voltage regulato
        status = "okay";                                /* enable USB HS PHY controller */
};

&usbphyc_port0 {
        st,phy-tuning = <&usb_phy_tuning>;              /* Optional USB HS PHY port#1 tuning *
};

&usbphyc_port1 {
        st,phy-tuning = <&usb_phy_tuning>;              /* Optional USB HS PHY port#2 tuning *
};
```

```
/ {                                                     /* Optional USB HS PHY tuning example,
        usb_phy_tuning: usb-phy-tuning {
                st,current-boost = <2>;
                st,no-lfs-fb-cap;
                st,hs-dc-level = <2>;
                st,hs-rftime-reduction;
                st,hs-current-trim = <5>;
                st,hs-impedance-trim = <0>;
                st,squelch-level = <1>;
                st,no-hs-ftime-ctrl;
                st,hs-tx-staggering;
        };
};
```

Static configuration of the UTMI switch to assign the **port#2** to either USBH or OTG is done by the **PHY user node**[5]:

- Please refer to USBH_device_tree_configuration
- Please refer to OTG_device_tree_configuration

usbphyc_port1 user must configure an additional specifier for UTMI switch: **0 to select OTG**, **1 to select USBH**

Abstract of the example to configure port#2, to be assigned to the USBH:

```
&usbh_ehci {
        phys = <&usbphyc_port0>, <&usbphyc_port1 1>; /* 1: UTMI switch selects the USBH */
        phy-names = "usb", "usb";
        ...
}
```

Abstract of the example to configure port#2, to be assigned to the OTG:

```
&usbotg_hs {
        phys = <&usbphyc_port1 0>;                  /* 0: UTMI switch selects the OTG */
        phy-names = "usb2-phy";
        ...
}
```

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. ↑ drivers/phy/st/phy-stm32-usbphyc.c , STM32 USB PHY Controller driver
2. ↑ [2.0] [2.1] Documentation/devicetree/bindings/phy/phy-stm32-usbphyc.txt , USBPHYC device tree bindings
3. ↑ arch/arm/boot/dts/stm32mp157c.dtsi , STM32MP157C device tree file
4. ↑ Regulator overview
5. ↑ Documentation/devicetree/bindings/phy/phy-bindings.txt ,PHY generic bindings