



HDP device tree configuration

HDP device tree configuration

Stable: 06.02.2020 - 14:37 / Revision: 06.02.2020 - 14:34

Contents

1 Article purpose	2
2 DT bindings documentation	2
3 DT configuration	2
3.1 DT configuration (STM32 level)	3
3.2 HDP DT configuration (board level)	3
3.3 DT configuration examples	3
4 How to configure the DT using STM32CubeMX	5
5 References	5

1 Article purpose

This article explains how to configure the HDP driver when the peripheral is assigned to the Linux[®]OS.

The configuration is performed using the device tree mechanism, which provides a hardware description of the Ethernet peripheral used by STM32 HDP driver

2 DT bindings documentation

The HDP tree bindings are composed of:

- STM32 HDP device tree bindings ^[1]



Please note that the upstreaming of the HDP Linux driver source code is in progress

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 HDP node is described in the `stm32mp157c.dtsi` ^[2] file with disabled status and required properties such as:

- The physical base address and size of the device register map
- The HDP clock

```
hdp: hdp@5002a000 {
    compatible = "st,stm32mp1-hdp";
    reg = <0x5002a000 0x400>;
    clocks = <&rcc HDP>;
    clock-names = "hdp";
    status = "disabled";
};
```

The required and optional properties are fully described in the `bindings` files.



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

3.2 HDP DT configuration (board level)

Part of the `device tree` describes the HDP hardware used on a given board. The DT node ("`hdp`") must be filled in as follows:

- Enable the HDP block by setting `status = "okay"`.
- Configure the pins in use via `pinctrl`, through `pinctrl-0` (default pins), `pinctrl-1` (sleep pins) and `pinctrl-names`.
- Configure the HDP interface using `muxing-hdp` to indicate which one of the 16 possible output pins is assigned to each HDP output.

```
&hdp {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&hdp_x_pins_y>;
    pinctrl-1 = <&hdp_x_pins_sleep_y>;
    status = "disabled";

    muxing-hdp = <(STM32_HDP(x, HDPx_value))>;
};
```

3.3 DT configuration examples

The example below shows how to configure and enable HDP instances at board level:

```
&hdp {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&hdp0_pins_a &hdp6_pins_a &hdp7_pins_a>;           /* configure
    pinctrl for hdp pin 0, 6 and 7*/
    pinctrl-1 = <&hdp0_pins_sleep_a &hdp6_pins_sleep_a &hdp7_pins_sleep_a>;
    status = "okay";                                               /* enable HDP
    */
```



HDP device tree configuration

```
    muxing-hdp = <(STM32_HDP(0, HDP0_GPOVAL_0) | /* For HDP
pin 0, the signal HDP0_GPOVAL_0 is selected*/
                STM32_HDP(6, HDP6_GPOVAL_6) | /* For HDP
pin 6, the signal HDP0_GPOVAL_6 is selected*/
                STM32_HDP(7, HDP7_GPOVAL_7))>; /* For HDP
pin 7, the signal HDP0_GPOVAL_7 is selected*/
};
```

List of all possible HDP signals:

```
/* define HDP Pins number*/
HDP0_PWR_PWRWAKE_SYS
HDP0_CM4_SLEEPDEEP
HDP0_PWR_STDBY_WKUP
HDP0_PWR_ENCOMP_VDDCORE
HDP0_BSEC_OUT_SEC_NIDEN
HDP0_RCC_CM4_SLEEPDEEP
HDP0_GPU_DBG7
HDP0_DDRCTRL_LP_REQ
HDP0_PWR_DDR_RET_ENABLE_N
HDP0_GPOVAL_0

HDP1_PWR_PWRWAKE_MCU
HDP1_CM4_HALTED
HDP1_CA7_NAXIERRIRQ
HDP1_PWR_OKIN_MR
HDP1_BSEC_OUT_SEC_DBGEN
HDP1_EXTI_SYS_WAKEUP
HDP1_RCC_PWRDS_MPU
HDP1_GPU_DBG6
HDP1_DDRCTRL_DFI_CTRLUPD_REQ
HDP1_DDRCTRL_CACTIVE_DDRC_ASR
HDP1_GPOVAL_1

HDP2_PWR_PWRWAKE_MPU
HDP2_CM4_RXEV
HDP2_CA7_NPMUIRQ1
HDP2_CA7_NFIQOUT1
HDP2_BSEC_IN_RSTCORE_N
HDP2_EXTI_C2_WAKEUP
HDP2_RCC_PWRDS_MCU
HDP2_GPU_DBG5
HDP2_DDRCTRL_DFI_INIT_COMPLETE
HDP2_DDRCTRL_PERF_OP_IS_REFRESH
HDP2_DDRCTRL_GSKP_DFI_LP_REQ
HDP2_GPOVAL_2

HDP3_PWR_SEL_VTH_VDD_CORE
HDP3_CM4_TXEV
HDP3_CA7_NPMUIRQ0
HDP3_CA7_NFIQOUT0
HDP3_BSEC_OUT_SEC_DFTLOCK
HDP3_EXTI_C1_WAKEUP
HDP3_RCC_PWRDS_SYS
HDP3_GPU_DBG4
HDP3_DDRCTRL_STAT_DDRC_REG_SELREF_TYPE0
HDP3_DDRCTRL_CACTIVE_1
HDP3_GPOVAL_3

HDP4_PWR_PDDS
HDP4_CM4_SLEEPING
HDP4_CA7_NRESET1
HDP4_CA7_NIRQOUT1
HDP4_BSEC_OUT_SEC_DFTEN
```



```
HDP4_BSEC_OUT_SEC_DBGSWENABLE
HDP4_ETH_OUT_PMT_INTR_0
HDP4_GPU_DBG3
HDP4_DDRCTRL_STAT_DDRC_REG_SELREF_TYPE1
HDP4_DDRCTRL_CACTIVE_0
HDP4_GPOVAL_4

HDP5_CA7_STANDBYWFIL2
HDP5_PWR_VTH_VDDCORE_ACK
HDP5_CA7_NRESET0
HDP5_CA7_NIRQOUT0
HDP5_BSEC_IN_PWROK
HDP5_BSEC_OUT_SEC_DEVICEEN
HDP5_ETH_OUT_LPI_INTR_0
HDP5_GPU_DBG2
HDP5_DDRCTRL_CACTIVE_DDRC
HDP5_DDRCTRL_WR_CREDIT_CNT
HDP5_GPOVAL_5

HDP6_CA7_STANDBYWFI1
HDP6_CA7_STANDBYWFE1
HDP6_CA7_EVENT0
HDP6_CA7_DBGACK1
HDP6_BSEC_OUT_SEC_SPNIDEN
HDP6_ETH_OUT_MAC_SPEED_01
HDP6_GPU_DBG1
HDP6_DDRCTRL_CSYSACK_DDRC
HDP6_DDRCTRL_LPR_CREDIT_CNT
HDP6_GPOVAL_6

HDP7_CA7_STANDBYWFI0
HDP7_CA7_STANDBYWFE0
HDP7_CA7_DBGACK0
HDP7_BSEC_OUT_FUSE_OK
HDP7_BSEC_OUT_SEC_SPIIDEN
HDP7_ETH_OUT_MAC_SPEED_00
HDP7_GPU_DBG0
HDP7_DDRCTRL_CSYSREQ_DDRC
HDP7_DDRCTRL_HPR_CREDIT_CNT
HDP7_GPOVAL_7
```

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

- [<Linux kernel directory>/Documentation/devicetree/bindings/soc/stm32/stm32_hdp.txt](#)



HDP device tree configuration

- `arch/arm/boot/dts/stm32mp157c.dtsi` , STM32MP157C device tree file

Operating System

Hardware Debug Port

Device Tree

Boot and Security and OTP control

Reset and Clock Control

Graphics Processing Units

Low Power (MIPI[®] Alliance DSI standard)

Doubledata rate (memory domain)

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

External Interrupt

Microprocessor Unit

Ethernet