FMC device tree configuration

FMC device tree configuration

Stable: 16.09.2019 - 14:27 / Revision: 16.09.2019 - 14:26

A quality version of this page, approved on *16 September 2019*, was based off this revision.
It was rated: **Expert**: Aproved **Technical writer**: Approved **Maintainer**: Approved

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the FMC peripheral, used by the STM32 FMC Linux driver and by the MTD framework.

2 DT bindings documentation

The FMC device tree bindings are composed of:

- generic MTD nand bindings [1].
- FMC driver bindings [2].

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 FMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
fmc: nand-controller@58002000 {                                 Comments
    compatible = "st,stm32mp15-fmc2";
    reg = <0x58002000 0x1000>,                                  --> First region contains the
          <0x80000000 0x1000>,                                  --> Régions 2 to 4 respective
          <0x88010000 0x1000>,
          <0x88020000 0x1000>,
          <0x81000000 0x1000>,                                  --> Régions 5 to 7 contain th
          <0x89010000 0x1000>,
          <0x89020000 0x1000>;
    interrupts = <GIC_SPI 48 IRQ_TYPE_LEVEL_HIGH>;              --> The interrupt number used
    dmas = <&mdma1 20 0x10 0x12000A02 0x0 0x0 0>,               --> DMA specifiers [4]
           <&mdma1 20 0x10 0x12000A08 0x0 0x0 0>,
           <&mdma1 21 0x10 0x12000A0A 0x0 0x0 0>;
    dma-names = "tx", "rx", "ecc";
    clocks = <&rcc FMC_K>;
    resets = <&rcc FMC_R>;
    status = "disabled";
};
```

This device tree part related to the STM32 should be kept as is, customer should not modify it.

3.2 DT configuration (board level)

The FMC peripheral may connect to one SLC NAND Flash memory (with a maximum of 2 dies per package).

```
&fmc {                                                          Comments
    pinctrl-names = "default", "sleep";                         --> For pinctrl configuration
    pinctrl-0 = <&fmc2_pins_a>;
    pinctrl-1 = <&fmc2_sleep_pins_a>;
    status = "okay";                                            --> Enable the node
    #address-cells = <1>;
    #size-cells = <0>;

    nand: nand@0 {
        reg = <0>;                                              --> Describe the CS line assi
        nand-on-flash-bbt;                                      --> Store the bad block table
        nand-ecc-strength = <8>;                                --> Number of bits to correct
        nand-ecc-step-size = <512>;                             --> Number of data bytes that
        #address-cells = <1>;
        #size-cells = <1>;
    };
};
```

The supported ECC strength and step size are:

- nand-ecc-strength = <1>, nand-ecc-step-size = <512> (HAMMING).
- nand-ecc-strength = <4>, nand-ecc-step-size = <512> (BCH4).
- nand-ecc-strength = <8>, nand-ecc-step-size = <512> (BCH8).

3.3 DT configuration examples

The below example shows how to configure the FMC controller when a SLC 8-bit NAND Flash memory device is connected (ECC requirement: 8 bits / 512 bytes).

```
&fmc {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&fmc2_pins_a>;
    pinctrl-1 = <&fmc2_sleep_pins_a>;
    status = "okay";
    #address-cells = <1>;
    #size-cells = <0>;

    nand: nand@0 {
        reg = <0>;
        nand-on-flash-bbt;
        #address-cells = <1>;
        #size-cells = <1>;

        partition@0 {
            ...
        };
    };
};
```

The below example shows how to configure the FMC controller when a SLC 8-bit NAND Flash memory device is connected (ECC requirement: 4 bits / 512 bytes).

```
&fmc {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&fmc2_pins_a>;
    pinctrl-1 = <&fmc2_sleep_pins_a>;
    status = "okay";
    #address-cells = <1>;
    #size-cells = <0>;

    nand: nand@0 {
        reg = <0>;
        nand-on-flash-bbt;
        nand-ecc-strength = <4>;
        nand-ecc-step-size = <512>;
        #address-cells = <1>;
        #size-cells = <1>;

        partition@0 {
            ...
        };
    };
};
```

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 full description:

- Documentation/devicetree/bindings/mtd/nand.txt
- Documentation/devicetree/bindings/mtd/stm32-fmc2-nand.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- Documentation/devicetree/bindings/dma/stm32-mdma.txt

Operating System

Memory Technology Device

Device Tree

Generic Interrupt Controller

Serial Peripheral Interface

Direct Memory Access

Single-Level Cell is a kind of NAND flash

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs.

Elliptic curve cryptography

Error Correction Capability