FMC device tree configuration

Contents

Stable: 15.4.2021 - 15.04 Revision: 15.4.2021 - 15.04

A quality version of this page, approved on *15 April 2021*, was based off this revision.

Contents

1 For ecosystem release v2.1.0

1.1 Article purpose

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

1.2 DT bindings documentation

The FMC device tree bindings are composed of:

- generic MTD NAND bindings [1].
- FMC NAND Flash controller driver bindings [2].
- FMC external bus interface driver bindings [3].

1.3 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.

1.3.1 DT configuration (STM32 level)

The FMC peripheral node is located in *stm32mp151.dtsi*[4] file.

```
   fmc: memory-controller@58002000 {                          Comments
       #address-cells = <2>;
       #size-cells = <1>;
       compatible = "st,stm32mp1-fmc2-ebi";
       reg = <0x58002000 0x1000>;                             --> This region contains the
register location
       clocks = <&rcc FMC_K>;
       resets = <&rcc FMC_R>;
       status = "disabled";

       ranges = <0 0 0x60000000 0x04000000>, /* EBI CS 1 */ --> External bus interface
region is used to address up to four external devices
                <1 0 0x64000000 0x04000000>, /* EBI CS 2 */
                <2 0 0x68000000 0x04000000>, /* EBI CS 3 */
                <3 0 0x6c000000 0x04000000>, /* EBI CS 4 */
                <4 0 0x80000000 0x10000000>; /* NAND */     --> NAND Flash controller
region is used  to address NAND Flash memory devices

       nand-controller@4,0 {
           #address-cells = <1>;
           #size-cells = <0>;
           compatible = "st,stm32mp1-fmc2-nfc";
           reg = <4 0x00000000 0x1000>,                       --> Regions 1 to 3
respectively contain the data, command and address space for CS0
```

```
                <4 0x08010000 0x1000>,
                <4 0x08020000 0x1000>,
                <4 0x01000000 0x1000>,                          --> Regions 4 to 6 contain
the same areas for CS1
                <4 0x09010000 0x1000>,
                <4 0x09020000 0x1000>;
            interrupts = <GIC_SPI 48 IRQ_TYPE_LEVEL_HIGH>;      --> The interrupt number used
            dmas = <&mdma1 20 0x2 0x12000a02 0x0 0x0 0x0>,      --> DMA specifiers [5]
                   <&mdma1 20 0x2 0x12000a08 0x0 0x0 0x0>,
                   <&mdma1 21 0x2 0x12000a0a 0x0 0x0 0x0>;
            dma-names = "tx", "rx", "ecc";
            status = "disabled";
        };
    };
```

Warning

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

1.3.2 DT configuration of the external bus interface controller (board level)

The FMC external bus interface controller may connect up to four external devices.

```
    &fmc {                                                      Comments
        pinctrl-names = "default", "sleep";                     --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
        pinctrl-0 = <&fmc2_pins_b>;
        pinctrl-1 = <&fmc2_sleep_pins_b>;
        status = "okay";                                        --> Enable the node

        ksz8851: ks8851mll@1,0 {
            compatible = "micrel,ks8851-mll";                   --> Configure the external
device
            reg = <1 0x0 0x2>, <1 0x2 0x20000>;
            interrupt-parent = <&gpioc>;
            interrupts = <3 IRQ_TYPE_LEVEL_LOW>;
            bank-width = <2>;

            st,fmc2-ebi-cs-mux-enable;                          --> Configure the
transactions with the external device
            st,fmc2-ebi-cs-transaction-type = <4>;
            st,fmc2-ebi-cs-buswidth = <16>;
            st,fmc2-ebi-cs-address-setup-ns = <5>;
            st,fmc2-ebi-cs-address-hold-ns = <5>;
            st,fmc2-ebi-cs-bus-turnaround-ns = <5>;
            st,fmc2-ebi-cs-data-setup-ns = <45>;
            st,fmc2-ebi-cs-data-hold-ns = <1>;
        };
    };
```

1.3.3 DT configuration of the NAND Flash controller (board level)

The FMC NAND Flash controller 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, please refer to Pinctrl device tree configuration
        pinctrl-0 = <&fmc2_pins_a>;
        pinctrl-1 = <&fmc2_sleep_pins_a>;
        status = "okay";                                        --> Enable the node
```

```
        nand-controller@4,0 {
            status = "okay";                                    --> Enable the NAND
controller node

            nand@0 {
                reg = <0>;                                      --> Describe the CS line
assigned to the NAND chip
                nand-on-flash-bbt;                              --> Store the bad block table
on NAND Flash memory
                nand-ecc-strength = <8>;                        --> Number of bits to correct
per ECC step
                nand-ecc-step-size = <512>;                     --> Number of data bytes that
are covered by a single ECC step
                #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).

Warning

It is recommended to check the ECC requirements in the datasheet of the memory provider.

1.3.4 DT configuration examples

The below example shows how to configure the FMC NAND Flash 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";

        nand-controller@4,0 {
            status = "okay";

            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 NAND Flash 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";

        nand-controller@4,0 {
            status = "okay";

            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 {
                    ...
                };
            };
        };
    };
```

1.4 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.

1.5 References

Please refer to the following links for full description:

- Documentation/devicetree/bindings/mtd/nand-controller.yaml (v5.4-stm32mp-r2)
- Documentation/devicetree/bindings/mtd/st,stm32-fmc2-nand.yaml (v5.4-stm32mp-r2)
- Documentation/devicetree/bindings/memory-controllers/st,stm32-fmc2-ebi.yaml (v5.4-stm32mp-r2)
- arch/arm/boot/dts/stm32mp151.dtsi (v5.4-stm32mp-r2)
- Documentation/devicetree/bindings/dma/stm32-mdma.txt (v5.4-stm32mp-r2)

2 For ecosystem release v2.0.0

2.1 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.

2.2 DT bindings documentation

The FMC device tree bindings are composed of:

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

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.

2.3.1 DT configuration (STM32 level)

The FMC peripheral node is located in *stm32mp151.dtsi*[3] file.

```
   fmc: nand-controller@58002000 {                                 Comments
       compatible = "st,stm32mp15-fmc2";
       reg = <0x58002000 0x1000>,                                  --> First region contains
the register location
             <0x80000000 0x1000>,                                  --> Regions 2 to 4
respectively contain the data, command and address space for CS0
             <0x88010000 0x1000>,
             <0x88020000 0x1000>,
             <0x81000000 0x1000>,                                  --> Regions 5 to 7 contain
the same areas for CS1
             <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";
   };
```

Warning

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

2.3.2 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, please refer to Pinctrl device tree 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
assigned to the NAND chip
           nand-on-flash-bbt;                                      --> Store the bad block
table on NAND Flash memory
           nand-ecc-strength = <8>;                                --> Number of bits to
correct per ECC step
           nand-ecc-step-size = <512>;                             --> Number of data bytes
that are covered by a single ECC step
           #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).

2.3.3 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 {
            ...
        };
    };
};
```

2.4 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.

2.5 References

Please refer to the following links for full description:

- Documentation/devicetree/bindings/mtd/nand-controller.yaml (v5.4-stm32mp-r1)
- Documentation/devicetree/bindings/mtd/stm32-fmc2-nand.txt (v5.4-stm32mp-r1)
- arch/arm/boot/dts/stm32mp151.dtsi (v5.4-stm32mp-r1)
- Documentation/devicetree/bindings/dma/stm32-mdma.txt (v5.4-stm32mp-r1)

Operating System

Memory Technology Device

Device Tree

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Generic Interrupt Controller

Serial Peripheral Interface

Direct Memory Access

Single-Level Cell is a kind of NAND flash

Elliptic curve cryptography

Error Correction Capability