FDCAN device tree configuration

Contents

Stable: 01.12.2020 - 10:33 / Revision: 10.06.2020 - 14:33

The content format pdf is not supported by the content model wikitext.

Return to Main Page.

Stable: 17.11.2021 - 16:46 / Revision: 17.11.2021 - 15:58

You do not have permission to edit this page, for the following reasons:

- The action you have requested is limited to users in one of the groups: Administrators, Editors, Reviewers, Selected_editors, ST_editors.
- The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

You can view and copy the source of this page.

== Article purpose == This article explains how to configure the [[FDCAN internal peripheral|FDCAN]] when it is assigned to the Linux<sup>&reg;</sup> OS. In that case, it is controlled by the [[CAN overview|CAN framework]] for Bosch M_CAN controller. The configuration is performed using the [[Device tree|device tree]] mechanism that provides a hardware description of the FDCAN peripheral, used by the M_CAN Linux driver and by the NET/CAN framework. If the peripheral is assigned to another execution context, refer to [[How to assign an internal peripheral to a runtime context]] article for guidelines on peripheral assignment and configuration. == DT bindings documentation == M_CAN device tree bindings<ref>{{CodeSource | Linux kernel | Documentation/devicetree /bindings/net/can/m_can.txt | Documentation/devicetree/bindings/net/can/m_can.txt}} M_CAN device tree bindings</ref> describe all the required and optional properties. == 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|How to configure the DT using STM32CubeMX]] for more details. ===DT configuration (STM32 level) === All M_CAN nodes are described in stm32mp157c.dtsi <ref>{{CodeSource | Linux kernel | arch/arm/boot/dts/stm32mp157c.dtsi | arch/arm/boot/dts/stm32mp157c.dtsi}}, STM32MP157C device tree file</ref> file with disabled status and required properties such as: * Physical base address and size of the device register map * Message RAM address and size (CAN SRAM) * Host clock and CAN clock * Message RAM configuration This is a set of properties that may not vary for a given STM32 device. m_can1: can@4400e000 { compatible = "bosch,m_can"; reg = <0x4400e000 0x400>, <0x44011000 0x1400>; {{highlight|/* FDCAN1 uses only the first half of the dedicated CAN_SRAM */}} reg-names = "m_can", "message_ram"; interrupts = <GIC_SPI 19 IRQ_TYPE_LEVEL_HIGH>, <GIC_SPI 21 IRQ_TYPE_LEVEL_HIGH>; interrupt-names = "int0", "int1"; clocks = <&rcc CK_HSE>, <&rcc FDCAN_K>; clock-names = "hclk", "cclk"; bosch,mram-cfg = <0x0 0 0 32 0 0 2 2>; status = "disabled"; }; m_can2: can@4400f000 { compatible = "bosch,m_can"; reg = <0x4400f000 0x400>, <0x44011000 0x2800>; {{highlight|/* The 10 Kbytes of the CAN_SRAM are mapped */}} reg-names = "m_can", "message_ram"; interrupts = <GIC_SPI 20 IRQ_TYPE_LEVEL_HIGH>, <GIC_SPI 22 IRQ_TYPE_LEVEL_HIGH>; interrupt-names = "int0", "int1"; clocks = <&rcc CK_HSE>, <&rcc FDCAN_K>; clock-names = "hclk", "cclk"; bosch,mram-cfg = <0x1400 0 0 32 0 0 2 2>; {{highlight|/* Set mram-cfg offset to write FDCAN2 data on the second half of the dedicated CAN_SRAM */}} status = "disabled"; }; The required and optional properties are fully described in the [[FDCAN_device_tree_configuration#DT_bindings_documentation|bindings files]]. {{Warning|This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.}} === DT configuration (board level) === Part of the [[Device tree|device tree]] is used to describe the FDCAN hardware used on a given board. The DT node (''"m_can"'') must be filled in: * Enable the CAN block by setting '''status = "okay".''' * Configure the pins in use via [[Pinctrl overview|pinctrl]], through '''pinctrl-0''' (default pins), '''pinctrl-1''' (sleep pins) and '''pinctrl-names'''. === DT configuration examples === The example below shows how to configure and enable FDCAN1 instance at board level: &m_can1 { pinctrl-names = "default", "sleep"; {{highlight|/* configure pinctrl modes for m_can1 */}} pinctrl-0 = <&m_can1_pins_a>; {{highlight|/* configure m_can1_pins_a as default pinctrl configuration for m_can1 */}} pinctrl-1 = <&m_can1_sleep_pins_a>; {{highlight|/* configure m_can1_sleep_pins_a as sleep pinctrl configuration for m_can1 */}} status = "okay"; {{highlight|/* enable m_can1 */}} }; ==How to configure the DT using STM32CubeMX== The [[STM32CubeMX]] tool can be used to configure the STM32MPU device and get the corresponding [[Device_tree#STM32|platform configuration device tree]] files.<br /> The STM32CubeMX may not support all the properties described in the above [[#DT bindings documentation|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. ==References== Please refer to the following links for additional information: <references /> <noinclude> [[Category:Device tree configuration]] [[Category:CAN|2]] {{ArticleBasedOnModel | Peripheral or framework device tree configuration model}} {{PublicationRequestId | 9832 | 2018-12-07 | AlainF}} </noinclude>

Templates used on this page:

- Template:Highlight (view source)
- Template:Info (view source)
- Template:STDarkBlue (view source)

Return to Main Page.