EXTI internal peripheral

Stable: 15.02.2019 - 16:11 / Revision: 15.02.2019 - 16:11

Contents

1 Peripheral overview

The **EXTI** peripheral is used to get an interrupt when a GPIO is toggling. It can also wake up the system from Stop low power mode, by means of the PWR internal peripheral when a wake up event occurs, before (eventualy - see the note below) propagating an interrupt to the client processor (Cortex-A7 GIC or Cortex-M4 NVIC). The wake up events can be internal (from other IPs clocked by the LSE, LSI or HSI from RCC), or external (from GPIO).

Notice that:

- Up to 16 GPIO pins can be configured as external interrupts: for each index between 0 and 15, one EXTI can be selected among all banks (EXTI<index> = GPIO<one_bank><index>).
- The 16 GPIO and 4 internal peripheral events can generate interrupts connected to the GIC and NVIC. All the other internal peripheral events can wake up the system, but the EXTI does not generate any interrupt to the GIC or NVIC for them; in such cases, another peripheral interrupt has to be used as a trigger via the GIC or NVIC.

1.1 Features

Refer to STM32MP15 reference manuals for the complete feature list, and to the software components, introduced below, to see which features are implemented.

1.2 Security support

The EXTI is a **secure** peripheral. By default, at reset, all EXTI wake up events are non-secure.

2 Peripheral usage and associated software

2.1 Boot time

The EXTI is not used by the boot chain, but is configured during Linux initialization. Since wake-up event configuration is done via register bit-field reads and writes, concurrent accesses from Linux and the co-processor are not possible at boot time:

- Linux configures all EXTI events during their respective consumer driver probing
- The co-processor uses the resource management mechanisms to request and configure the EXTI events it needs.

The article STM32MP15 interrupts gives more information on the EXTI configuration strategy.

2.2 Runtime

2.2.1 Overview

The EXTI can be allocated to:

- the Cortex-A7 non-secure for use in Linux with the interrupts framework

or

- the Cortex-M4 for using in STM32Cube with the EXTI HAL driver

2.2.2 Software frameworks

Domain	Peripheral	Software frameworks			Comment
		Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Interrupts	EXTI		Linux interrupt framework	STM32Cube EXTI driver	

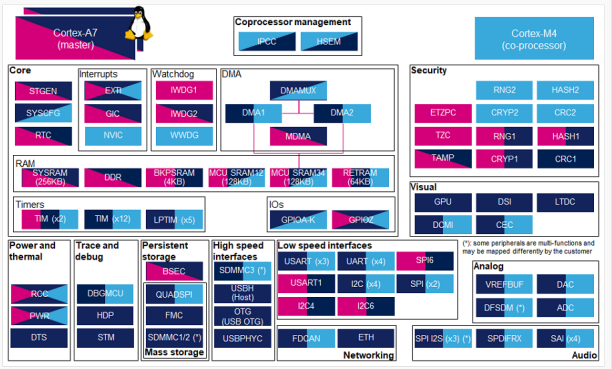
2.2.3 Peripheral configuration

The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (particularly for external peripherals), according to the information given in the corresponding software framework article.

2.2.4 Peripheral assignment

Check boxes illustrate the possible peripheral allocations supported by STM32 MPU Embedded Software:

- ☐ means that the peripheral can be assigned (☑) to the given runtime context.
- ✓ is used for system peripherals that cannot be unchecked because they are statically connected in the device.



Internal peripherals

Refer to How to assign an internal peripheral to a runtime context for more information on how to assign peripherals manually or via STM32CubeMX.
The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possiblities might be described in STM32MP15 reference manuals.

Domain	Peripheral	Runtime allocation				Comment
		Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Interrupts	EXTI	EXTI		☐	☐	Shareable (multiple choices supported)