Category:IOs pins peripherals

Contents

Stable: 17.06.2020 - 15:20 / Revision: 16.01.2020 - 12:31

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to the **IOs pins** internal peripherals (hardware blocks) embedded in the STM32 MPUs microprocessor devices.

Pages in category "IOs pins peripherals"

This category contains only the following page.

- GPIO internal peripheral

Stable: 19.11.2020 - 08:26 / Revision: 19.11.2020 - 08:24

A quality version of this page, approved on *19 November 2020*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to

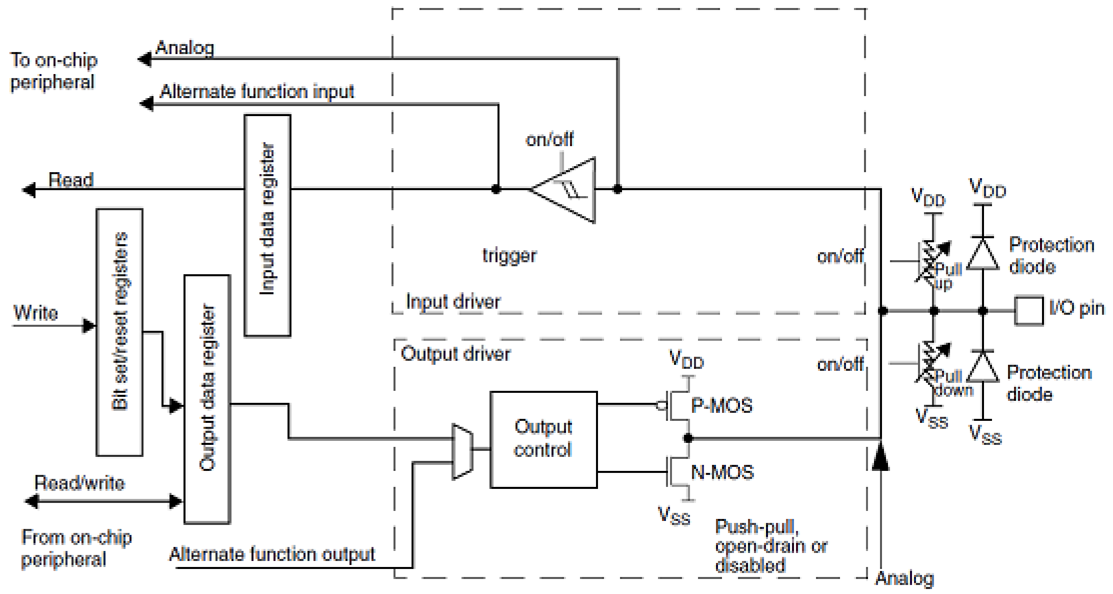
- briefly introduce the GPIO peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain how to configure the GPIO peripheral.

2 Peripheral overview

The **GPIO** peripheral is used to configure the device IO ports, also called pins or pads.
Each GPIO instance controls 8 pins (for GPIOK and GPIOZ) or 16 pins (for GPIOA to GPIOJ).

Every IO port implements the logic shown in the image below, taken from STM32MP15 reference manuals:

- The **IO pin** (on the right) is the physical connection to a chip external ball, soldered on the PCB. The link between each GPIO pin and each ball of the package is given in the datasheet.
- The **Read** and **Write** accesses allow the processor (Arm®Cortex®-A7 or Arm®Cortex®-M4) to configure the peripheral, control the IO pin and get its status.
- **Alternate function** (AF) links allow to connect the IO port to an internal peripheral digital line. In such a case, the IO direction is given by the line purpose: for instance, UART transmit line (TX) is an output.
- **Analog** links allow to connect the IO port to an internal peripheral analog line. In such a case, the IO direction is given by the line purpose: for instance, ADC input line is an input.



Note:

- the pull-up and pull-down resistors are disabled (by hardware) in analog mode.
- at reset, all pins are set in analog input mode to protect the device and minimize the power consumption. All unused pins should be kept in this state.

The pin configuration done by the software consists in:

- setting the **pin mode** in the GPIOx_MODER register:
  - **input** or **output** if the pin is used as general purpose (GPIO), controlled by software.
  - **analog**.
  - **alternate function** (AF).
- selecting the **alternate function** in the GPIOx_AFRH/L register (only when the pin mode is AF):
  - each IO port can support up to 16 alternate functions that are documented in the datasheet.

- setting the **pin characteristics**:
  - **no pull-up and no pull-down** or **pull-up** or **pull-down** in the GPIOx_PUPDR register, needs to be selected to be coherent with the hardware schematics.
  - **push-pull** or **open-drain** in the GPIOx_OTYPER register, needs to be selected to be coherent with the hardware schematics.
  - **output speed** in the GPIOx_OSPEEDR register needs to be tuned to achieve the expected level of performance (rising and falling times) while limiting electromagnetic interferences (EMI) and overconsumption. As example, the table below summarizes the maximum achievable frequency for each supported IO voltage and a 30pF load:

GPIOx_OSPEEDR	Meaning	VDD=3v3	VDD=1v8 HSLV OFF	VDD=1v8 HSLV ON
b00	Low speed	24 MHz	11 MHz	22 MHz
b01	Medium speed	83 MHz	28 MHz	79 MHz
b10	High speed	125 MHz	66 MHz	101 MHz
b11	Very high speed	150 MHz	70 MHz	111 MHz

Note:

- More information is available in the **IO speed settings** chapter of the AN5031[1].
- There are different **IO types** with different characteristics: for instance, all pads are not able to achieve 150 MHz while supplied at 3v3. Refer to the datasheet to get the characteristics for each pin.
- When supplied with VDD=1v8, it is possible to enable the **high speed low voltage** (HSLV) pad mode for FTH (Five volt Tolerant High speed) and FTE (Five volt Tolerant Extended high speed) IO types on some peripherals (SPI, SDMMC, ETH , Dual QUADSPI and TRACE) via SYSCFG_IOCTRLR register in SYSCFG. **Warning**: As it could be destructive if used when VDD>2.7V, the HSLVEN_xx bits set in SYSCFG_IOCTRLR register are not taken account unless the OTP bit PRODUCT_BELOW_2V5 is set.

The table below shows all possible characteristics combinations for each **pin mode**:

pin mode	GPIOx_PUPDR	GPIOx_OTYPER	GPIOx_OSPEEDR
analog	Not applicable	Not applicable	Not applicable
input (GPIO or AF)	**no pull-up and no pull-down**	Not applicable	Not applicable
output (GPIO or AF) or **bi-**			

pin mode	GPIOx_PUPDR	GPIOx_OTYPER	GPIOx_OSPEEDR
directional (AF)	or **pull-down** or **pull-up**	**push-pull** or **open-drain**	cf. the table above

Note:

- 'Not applicable' means that setting this register has no effect but, in any case, there is no risk for the device.
- On the other hand, leaving a register not initialized whereas it should be, may lead to an unpredictable behavior!

2.1 Features

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

2.2 Security support

The GPIOA to GPIOK peripherals are **non-secure**.
The GPIOZ peripheral is **secure aware**.

3 Peripheral usage and associated software

3.1 Boot time

The STM32CubeMX tool allows to configure in one place the GPIO configuration that is applied at boot time and used at runtime, so it is highly recommended to use it to generate your device tree. Moreover, STM32CubeMX integrates all the information documented in the datasheet, making this configuration step straightforward.

Since a GPIO configuration is done via atomic registers read and write, concurrent accesses from different cores must be avoided and that is why all GPIO configurations are done by the Arm®Cortex®-A7. The strategy is to progressively initialize the GPIO all along the boot chain, as soon as one boot component needs to use them:

- Most of the GPIOs used by the ROM code are directly defined in the ROM code but it is possible to change some pins muxing via dedicated words in BSEC.
- The other boot components are relying on a common binding[2] in the device tree to get the pins configuration:
  - The FSBL configures both secure (GPIOZ) and non-secure (GPIOA to GPIOK) instances.
  - The SSBL and Linux pinctrl configure only non-secure instances, so GPIOA to GPIOK, and only the pins left non-secure in GPIOZ by the FSBL. Linux also initializes the GPIO used by the coprocessor, via its resource manager.

3.2 Runtime

3.2.1 Overview

The **GPIO configuration** must not be done from different cores to avoid concurrent accesses, but this is not the case for the **GPIO using**: each core can manipulate IO on its own since dedicated set/clear registers are available for that.

Nevertheless, beyond the boot time, the GPIO configuration also evolves at runtime: while entering in low power mode, some GPIOs may be put back to analog input mode in order to reduce the power consumption. This is done in two times:

1. the Arm®Cortex®-A7 non-secure takes care of the non-secure GPIO with Linux IOs pins frameworks.
2. the Arm®Cortex®-A7 secure takes care of the secure pins of GPIOZ behind PSCI secure services.

On wakeup, the boot chain restores the GPIO configuration similarly to what is done at boot time.

Let's come back to the runtime allocation...

The GPIOZ pins can individually be:

- set secure and assigned to the Arm®Cortex®-A7 secure for using with OP-TEE

or

- set non-secure and assigned to the Arm®Cortex®-A7 non-secure for using in Linux® with the IOs pins frameworks

or

- set non-secure and assigned to the Arm®Cortex®-M4 for using in STM32Cube with GPIO HAL driver

The GPIOA to GPIOK pins can individually be:

- assigned to the Arm®Cortex®-A7 non-secure for using in Linux® with the IOs pins frameworks or

- assigned to the Arm®Cortex®-M4 for using in STM32Cube with GPIO HAL driver

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks			Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/IOs	GPIO	OP-TEE GPIO driver	Linux IOs pins overview	STM32Cube GPIO driver	

3.2.3 Peripheral configuration

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

In Linux kernel, each GPIO bank is declared as a "gpio-controller" in the device tree and each pin can then be used via two different consumer frameworks:

- Pinctrl framework is used to control the alternate function (AF) selection for a given device driver, via the Pinctrl device tree configuration.
- Gpiolib framework is used to control a pin in GPIO mode from another device driver or a user space application: refer to GPIO device tree configuration for further details.

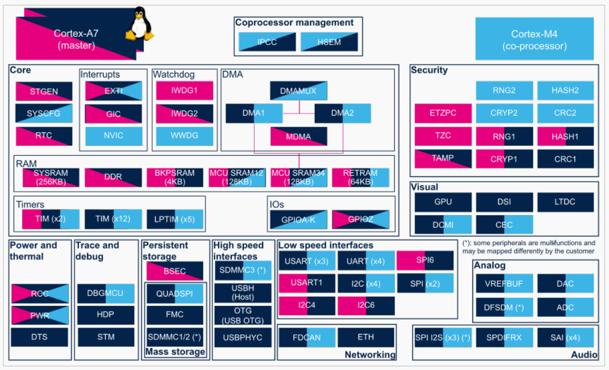
3.2.4 Peripheral assignment

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.

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
.



Internal peripherals

Domain	Periphera l	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cub e)		

Domain	Peripheral	Runtime allocation				Comment
Core/IOs	GPIO	GPIOA (16 pins)				Shareable (with pin granularity)
		GPIOB (16 pins)				Shareable (with pin granularity)
		GPIOC (16 pins)				Shareable (with pin granularity)
		GPIOD (16 pins)				Shareable (with pin granularity)
		GPIOE (16 pins)				Shareable (with pin granularity)
		GPIOF (16 pins)				Shareable (with pin granularity)
		GPIOG (16 pins)				Shareable (with pin granularity)
		GPIOH (16 pins)				Shareable (with pin granularity)
		GPIOI (16 pins)				Shareable (with pin granularity)
		GPIOJ (16 pins)				Shareable (with pin granularity)
		GPIOK (8 pins)				Shareable (with pin granularity)
		GPIOZ (8 pins)				Shareable (with pin granularity)

4 How to go further

Not applicable.

5 References

- AN5031
- Documentation/devicetree/bindings/pinctrl/st,stm32-pinctrl.yaml

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

input/output

Printed Circuit Board

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

GPIO alternate function

Transmit

External memory interface

High Speed Low Voltage pin mode

System Configuration

One Time Programmed

Read Only Memory

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment