ADC internal peripheral

Contents

Stable: 11.02.2019 - 11:20 / Revision: 18.01.2019 - 16:01

Stable: 11.02.2019 - 11:24 | Revision: 18.01.2019 - 10:01

A quality version of this page, accepted on *11 February 2019*, was based off this revision.

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

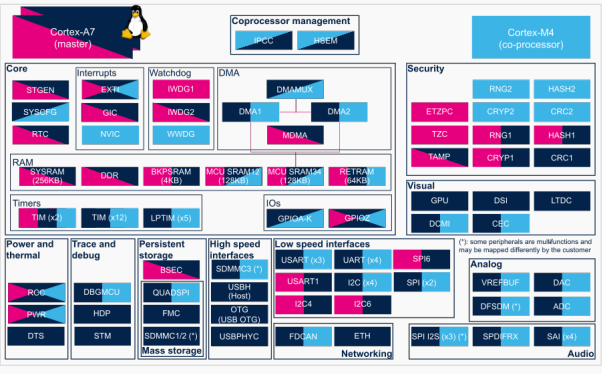
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 16.01.2020 - 15:02 / Revision: 16.01.2020 - 14:59

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

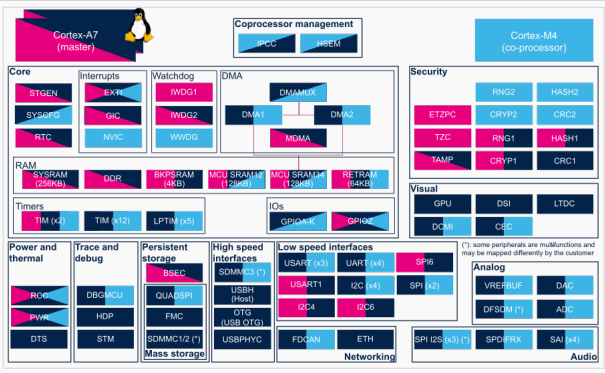
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 26.03.2021 - 13:52 / Revision: 25.01.2021 - 09:04

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

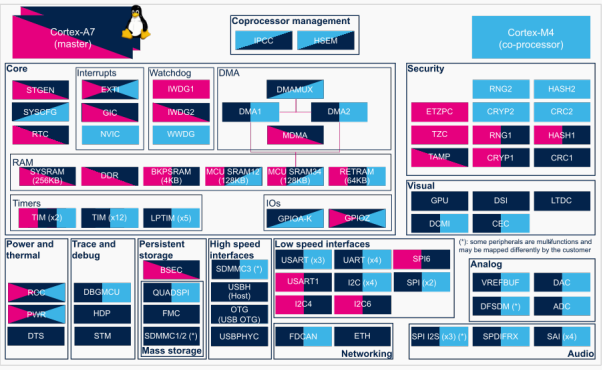
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 25.09.2020 - 08:36 / Revision: 25.09.2020 - 08:35

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

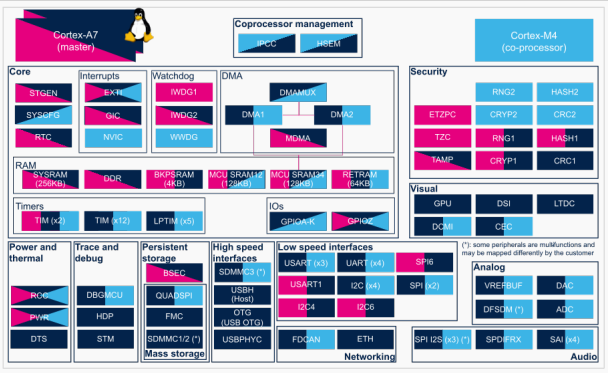
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 13.10.2020 - 08:29 / Revision: 13.10.2020 - 08:29

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

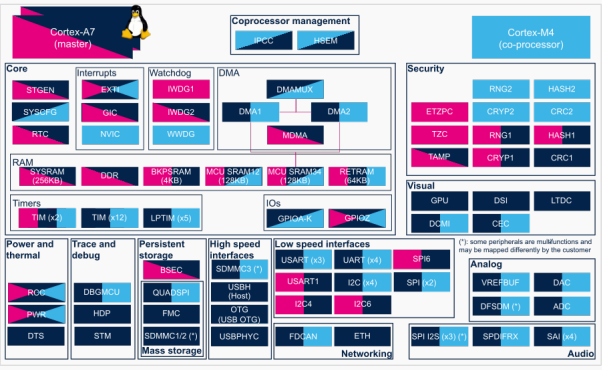
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 26.03.2021 - 13:10 / Revision: 18.03.2021 - 14:45

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

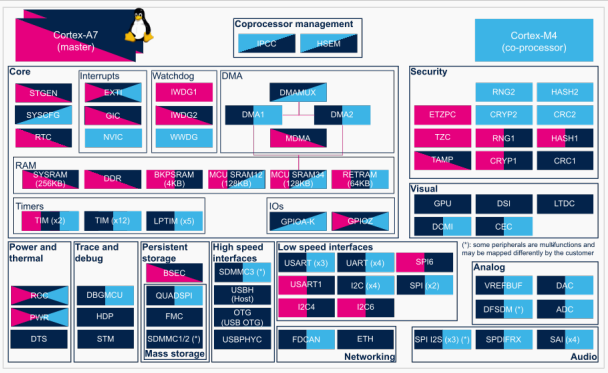
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 08.03.2021 - 16:13 / Revision: 16.02.2021 - 17:11

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

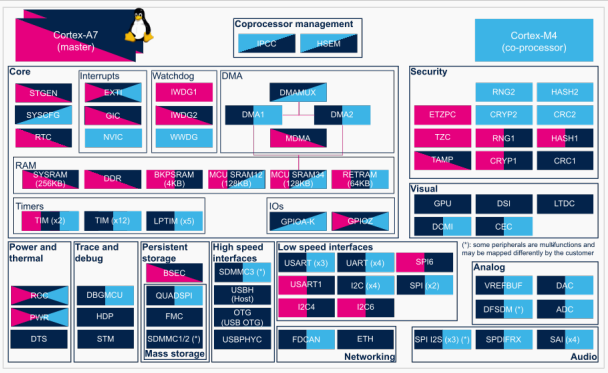
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 17.02.2021 - 16:24 / Revision: 17.02.2021 - 16:22

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

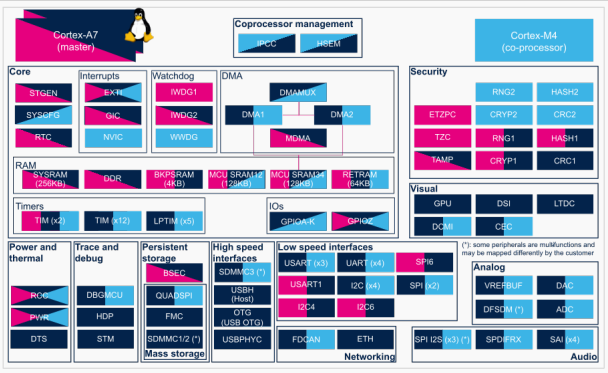
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 30.03.2021 - 13:25 / Revision: 29.03.2021 - 09:46

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

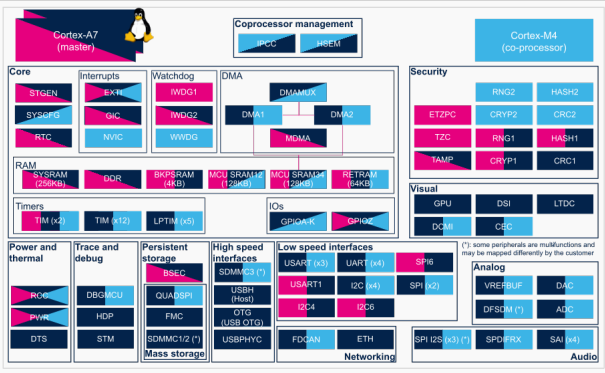
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 25.09.2020 - 09:10 / Revision: 25.09.2020 - 09:09

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

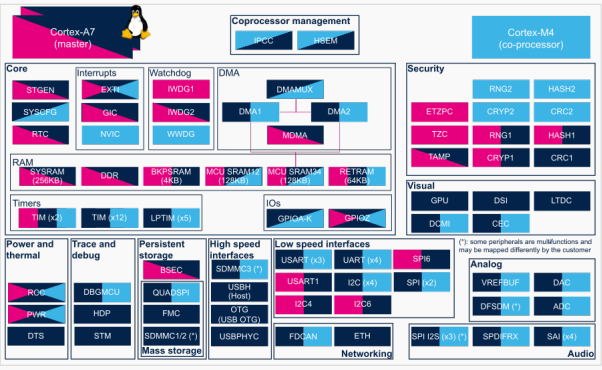
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

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

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 11.06.2020 - 12:50 / Revision: 11.06.2020 - 12:12

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

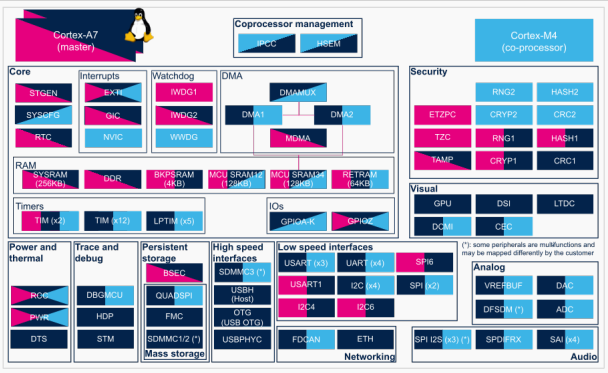
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 31.03.2021 - 11:58 / Revision: 23.03.2021 - 14:07

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

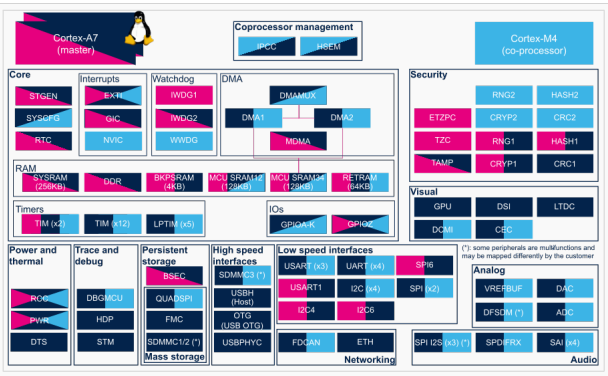
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 23.09.2020 - 13:22 / Revision: 12.06.2020 - 13:25

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

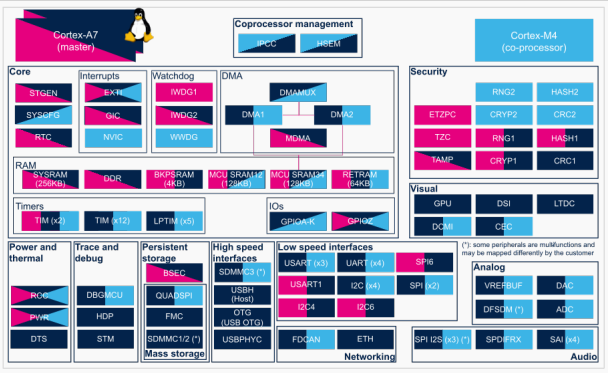
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: Not stable / Revision: 11.03.2021 - 08:07

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

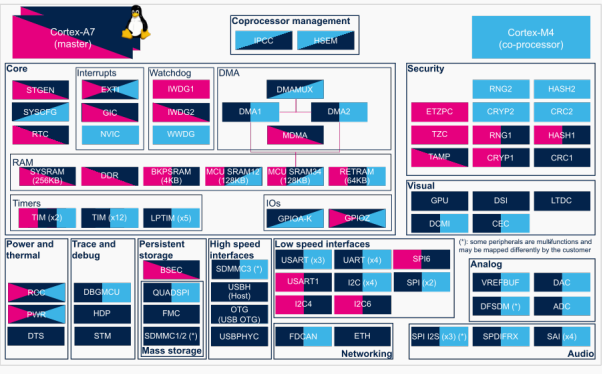
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 26.03.2021 - 11:32 / Revision: 12.03.2021 - 11:07

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

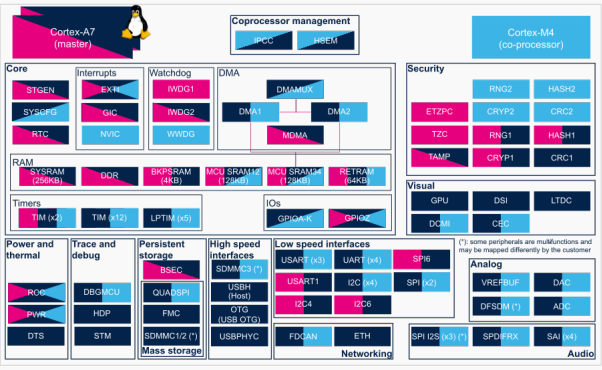
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Cortex®

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 30.03.2021 - 13:21 / Revision: 29.03.2021 - 09:46

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

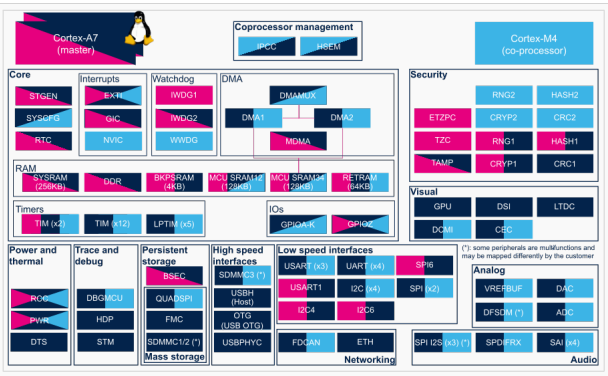
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 25.09.2020 - 09:43 / Revision: 25.09.2020 - 09:37

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the ADC 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 ADC peripheral.

2 Peripheral overview

The STM32 **ADC** is a successive approximation analog-to-digital converter.

2.1 Features

The STM32MP15 has one ADC block with two physical ADCs:

- **Configurable resolution**: 8, 10, 12, 14, 16 bits.
- Each ADC has up to 20 **multiplexed channels** (including 6 internal channels connected only to ADC2).
- The conversions can be performed in **single, continuous, scan or discontinuous mode**.
- The result can be read in a left- or right-aligned 32-bit data register by using **CPU or DMA**[1].
- The **analog watchdog** feature allows the application to detect if the input voltage goes beyond the user-defined, high or low thresholds.
- A **common input clock** for the two ADCs, which can be selected between 2 different clock[2] sources (Synchronous or Asynchronous clock).
- The **common reference voltage** can be provided by either VREFBUF[3] or any other external regulator[4] wired to VREF+ pin.

Each ADC supports two contexts to manage conversions:

- **Regular conversions** can be done in sequence, running in background
- **Injected conversions** have higher priority, and so have the ability to interrupt the regular sequence (either triggered in SW or HW). The regular sequence is resumed, in case it has been interrupted.
- Each context has its own **configurable sequence and trigger**: software, TIM[5], LPTIM[6] and EXTI[7].

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 ADC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The ADC is usually not used at boot time. But it may be used by the SSBL (see Boot chains overview), to check for power supplies for example.

3.2 Runtime

3.2.1 Overview

The ADC can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with IIO framework.

or

- the Arm®Cortex®-M4 to be used with STM32Cube MPU Package with ADC HAL driver.

The Peripheral assignment chapter describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Analog	ADC		IIO framework	STM32Cube ADC driver	

3.2.3 Peripheral configuration

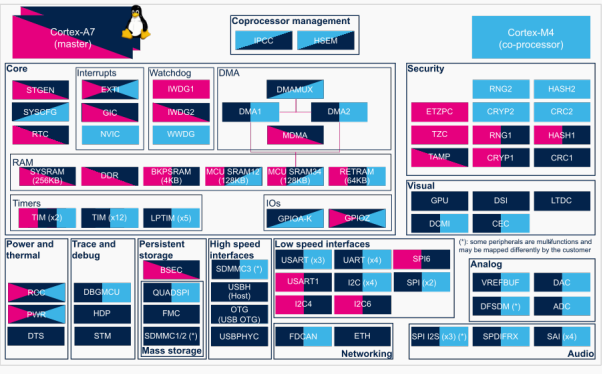
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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

For the Linux kernel configuration, please refer to ADC device tree configuration and ADC Linux driver articles.

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.



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)			
Analog	ADC	ADC				Assignment (single choice)

4 How to go further

See the application note:

- How to get the best ADC accuracy in STM32[8].

5 References

- DMA internal peripheral
- RCC internal peripheral
- VREFBUF internal peripheral
- Regulator overview
- TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- How to get the best ADC accuracy in STM32, by STMicroelectronics

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Central processing unit

Direct Memory Access

voltage reference buffer (STM32 specific)

low-power timer (STM32 specific)

External Interrupt

Second Stage Boot Loader

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

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit

Open Portable Trusted Execution Environment