



ADC internal peripheral



Contents

1. ADC internal peripheral	3
2. ADC Linux driver	9
3. ADC device tree configuration	16
4. Boot chains overview	23
5. DMA internal peripheral	30
6. EXTI internal peripheral	37
7. How to assign an internal peripheral to a runtime context	44
8. IIO overview	51
9. LPTIM internal peripheral	58
10. RCC internal peripheral	65
11. Regulator overview	72
12. STM32CubeMP1 architecture	79
13. STM32CubeMX	86
14. STM32MP15 resources	93
15. STM32MPU Embedded Software architecture overview	100
16. TIM internal peripheral	107
17. VREFBUF internal peripheral	114



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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	4
2 Peripheral overview	5
2.1 Features	5
2.2 Security support	5
3 Peripheral usage and associated software	6
3.1 Boot time	6
3.2 Runtime	6
3.2.1 Overview	6
3.2.2 Software frameworks	6
3.2.3 Peripheral configuration	6
3.2.4 Peripheral assignment	6
4 How to go further	8
5 References	9



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

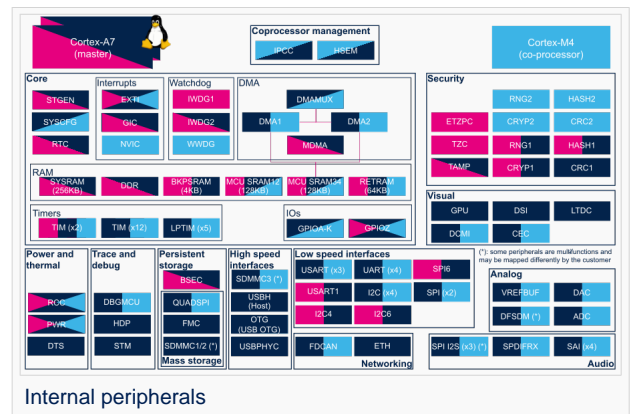
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	11
2 Peripheral overview	12
2.1 Features	12
2.2 Security support	12
3 Peripheral usage and associated software	13
3.1 Boot time	13
3.2 Runtime	13
3.2.1 Overview	13
3.2.2 Software frameworks	13
3.2.3 Peripheral configuration	13



3.2.4 Peripheral assignment	13
4 How to go further	15
5 References	16



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

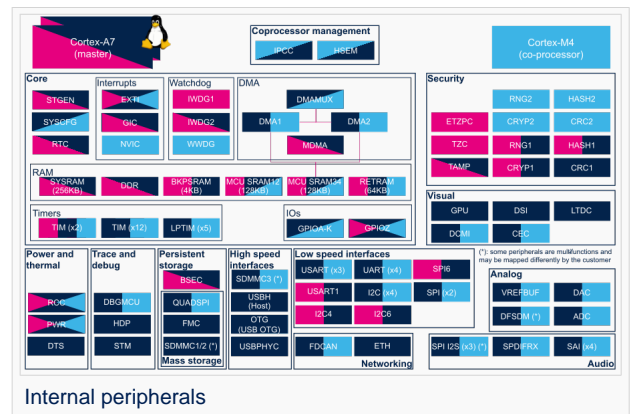
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	18
2 Peripheral overview	19
2.1 Features	19
2.2 Security support	19
3 Peripheral usage and associated software	20
3.1 Boot time	20
3.2 Runtime	20
3.2.1 Overview	20
3.2.2 Software frameworks	20
3.2.3 Peripheral configuration	20



3.2.4 Peripheral assignment	20
4 How to go further	22
5 References	23



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

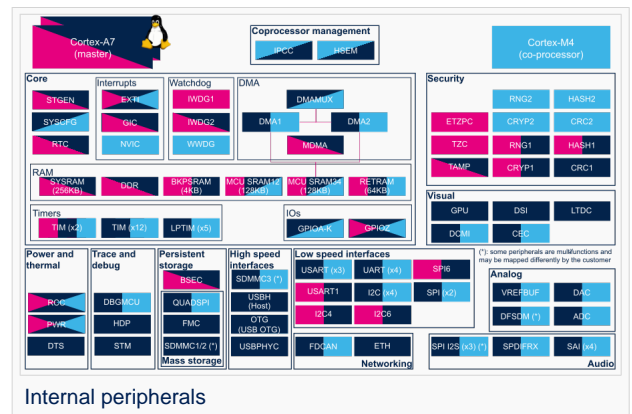
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	25
2 Peripheral overview	26
2.1 Features	26
2.2 Security support	26
3 Peripheral usage and associated software	27
3.1 Boot time	27
3.2 Runtime	27
3.2.1 Overview	27
3.2.2 Software frameworks	27
3.2.3 Peripheral configuration	27



3.2.4 Peripheral assignment	27
4 How to go further	29
5 References	30



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

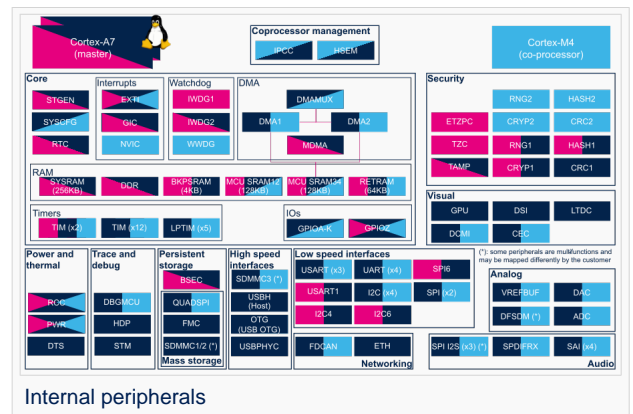
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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 possibilities 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: 13.10.2020 - 08:29 / Revision: 13.10.2020 - 08:29

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	32
2 Peripheral overview	33
2.1 Features	33
2.2 Security support	33
3 Peripheral usage and associated software	34
3.1 Boot time	34
3.2 Runtime	34
3.2.1 Overview	34
3.2.2 Software frameworks	34
3.2.3 Peripheral configuration	34



3.2.4 Peripheral assignment	34
4 How to go further	36
5 References	37



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

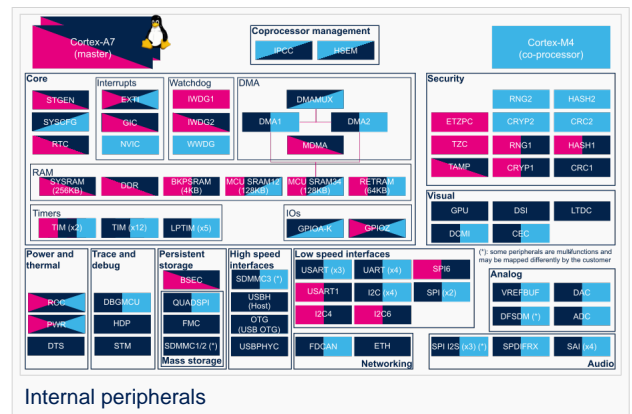
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	39
2 Peripheral overview	40
2.1 Features	40
2.2 Security support	40
3 Peripheral usage and associated software	41
3.1 Boot time	41
3.2 Runtime	41
3.2.1 Overview	41
3.2.2 Software frameworks	41
3.2.3 Peripheral configuration	41



3.2.4 Peripheral assignment	41
4 How to go further	43
5 References	44



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

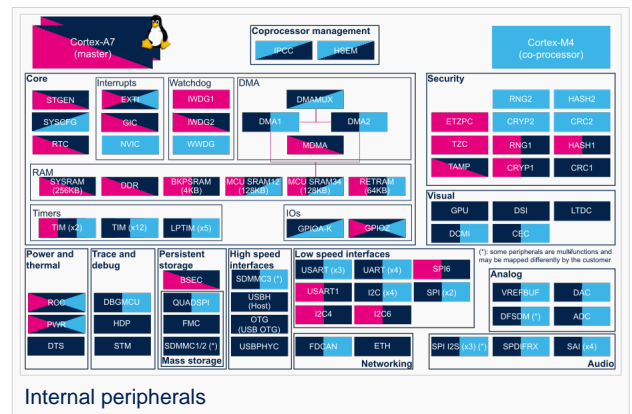
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	46
2 Peripheral overview	47
2.1 Features	47
2.2 Security support	47
3 Peripheral usage and associated software	48
3.1 Boot time	48
3.2 Runtime	48
3.2.1 Overview	48
3.2.2 Software frameworks	48
3.2.3 Peripheral configuration	48



3.2.4 Peripheral assignment	48
4 How to go further	50
5 References	51



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

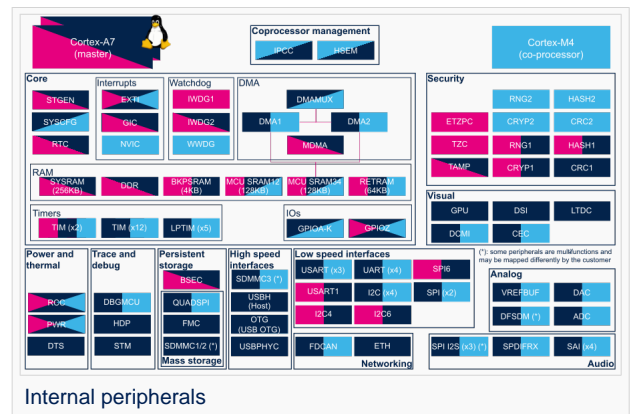
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	53
2 Peripheral overview	54
2.1 Features	54
2.2 Security support	54
3 Peripheral usage and associated software	55
3.1 Boot time	55
3.2 Runtime	55
3.2.1 Overview	55
3.2.2 Software frameworks	55
3.2.3 Peripheral configuration	55



3.2.4 Peripheral assignment	55
4 How to go further	57
5 References	58



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

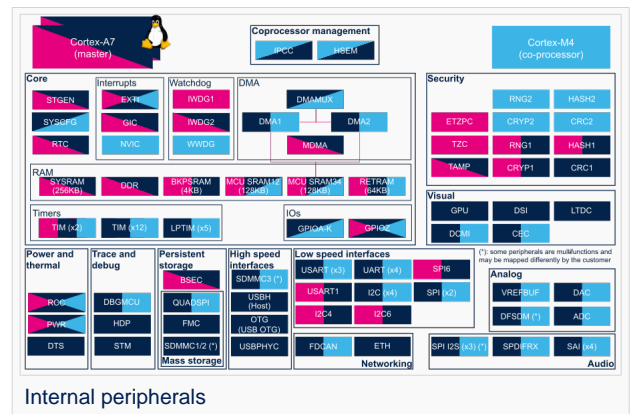
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	60
2 Peripheral overview	61
2.1 Features	61
2.2 Security support	61
3 Peripheral usage and associated software	62
3.1 Boot time	62
3.2 Runtime	62
3.2.1 Overview	62
3.2.2 Software frameworks	62
3.2.3 Peripheral configuration	62



3.2.4 Peripheral assignment	62
4 How to go further	64
5 References	65



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

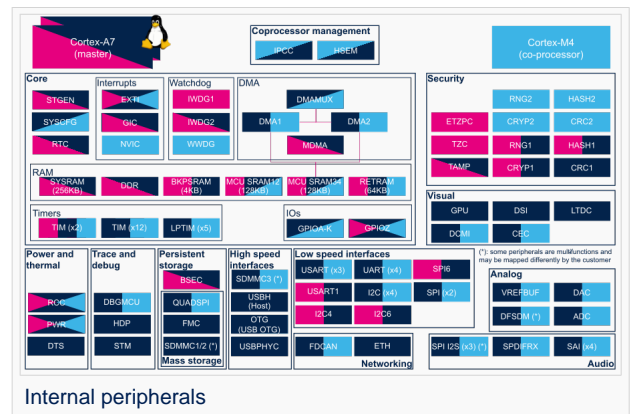
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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 possibilities 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: 25.09.2020 - 09:10 / Revision: 25.09.2020 - 09:09

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	67
2 Peripheral overview	68
2.1 Features	68
2.2 Security support	68
3 Peripheral usage and associated software	69
3.1 Boot time	69
3.2 Runtime	69
3.2.1 Overview	69
3.2.2 Software frameworks	69
3.2.3 Peripheral configuration	69



3.2.4 Peripheral assignment	69
4 How to go further	71
5 References	72



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

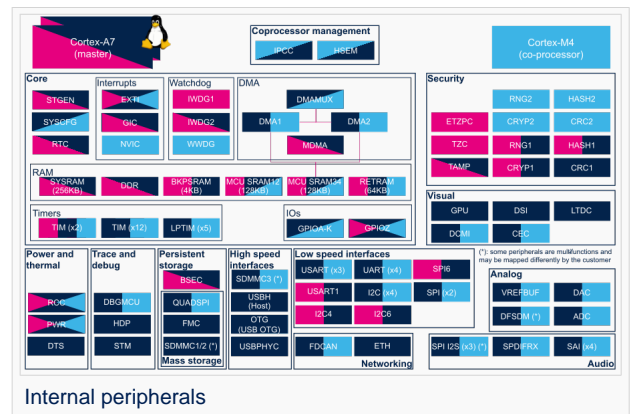
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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 possibilities 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	74
2 Peripheral overview	75
2.1 Features	75
2.2 Security support	75
3 Peripheral usage and associated software	76
3.1 Boot time	76
3.2 Runtime	76
3.2.1 Overview	76
3.2.2 Software frameworks	76
3.2.3 Peripheral configuration	76



3.2.4 Peripheral assignment	76
4 How to go further	78
5 References	79



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

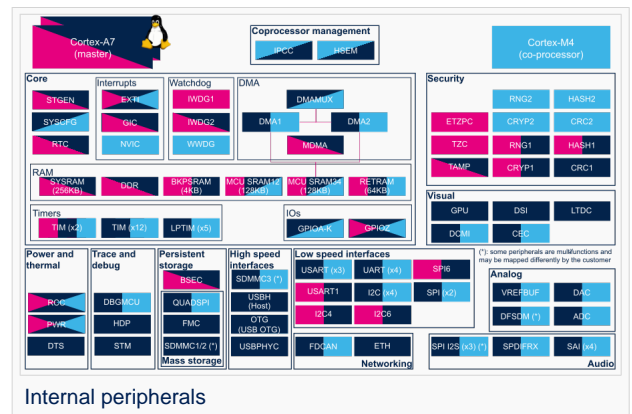
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	81
2 Peripheral overview	82
2.1 Features	82
2.2 Security support	82
3 Peripheral usage and associated software	83
3.1 Boot time	83
3.2 Runtime	83
3.2.1 Overview	83
3.2.2 Software frameworks	83
3.2.3 Peripheral configuration	83



3.2.4 Peripheral assignment	83
4 How to go further	85
5 References	86



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

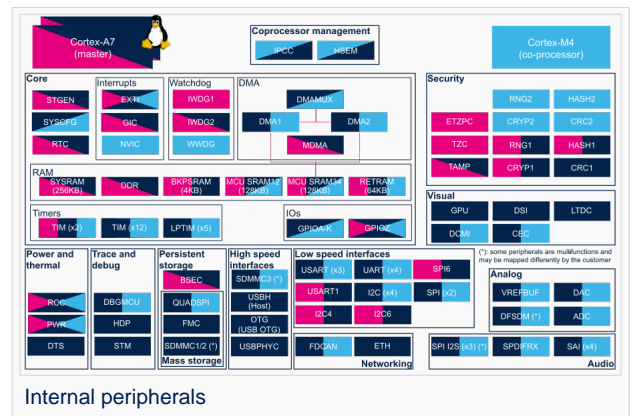
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	88
2 Peripheral overview	89
2.1 Features	89
2.2 Security support	89
3 Peripheral usage and associated software	90
3.1 Boot time	90
3.2 Runtime	90
3.2.1 Overview	90
3.2.2 Software frameworks	90
3.2.3 Peripheral configuration	90



3.2.4 Peripheral assignment	90
4 How to go further	92
5 References	93



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

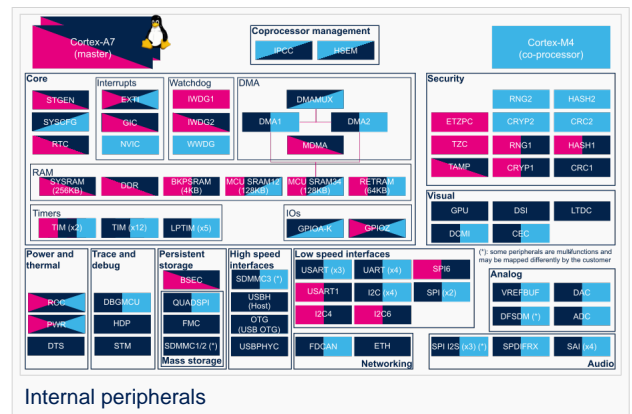
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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 possibilities 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: **Not stable** / Revision: 11.03.2021 - 08:07

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	95
2 Peripheral overview	96
2.1 Features	96
2.2 Security support	96
3 Peripheral usage and associated software	97
3.1 Boot time	97
3.2 Runtime	97
3.2.1 Overview	97
3.2.2 Software frameworks	97
3.2.3 Peripheral configuration	97



3.2.4 Peripheral assignment	97
4 How to go further	99
5 References	100



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

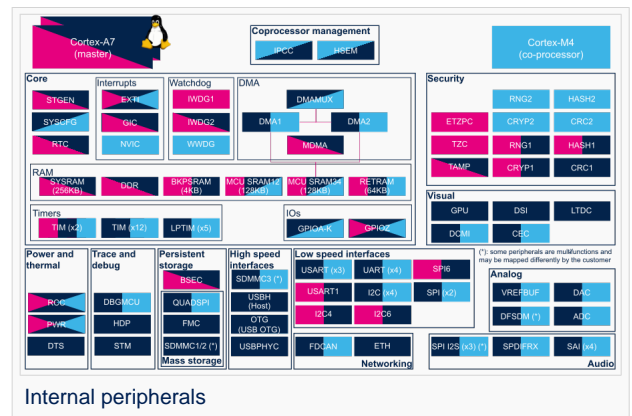
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	102
2 Peripheral overview	103
2.1 Features	103
2.2 Security support	103
3 Peripheral usage and associated software	104
3.1 Boot time	104
3.2 Runtime	104
3.2.1 Overview	104
3.2.2 Software frameworks	104
3.2.3 Peripheral configuration	104



3.2.4 Peripheral assignment	104
4 How to go further	106
5 References	107



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

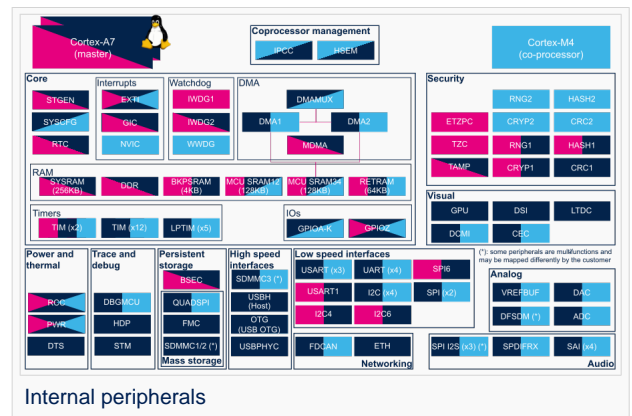
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	109
2 Peripheral overview	110
2.1 Features	110
2.2 Security support	110
3 Peripheral usage and associated software	111
3.1 Boot time	111
3.2 Runtime	111
3.2.1 Overview	111
3.2.2 Software frameworks	111
3.2.3 Peripheral configuration	111



3.2.4 Peripheral assignment	111
4 How to go further	113
5 References	114



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

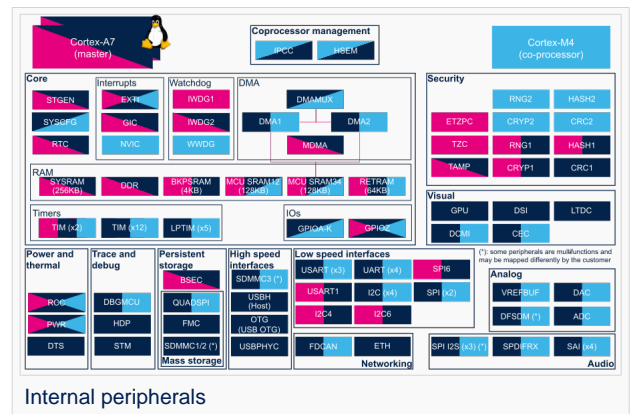
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	116
2 Peripheral overview	117
2.1 Features	117
2.2 Security support	117
3 Peripheral usage and associated software	118
3.1 Boot time	118
3.2 Runtime	118
3.2.1 Overview	118
3.2.2 Software frameworks	118
3.2.3 Peripheral configuration	118



3.2.4 Peripheral assignment	118
4 How to go further	120
5 References	121



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

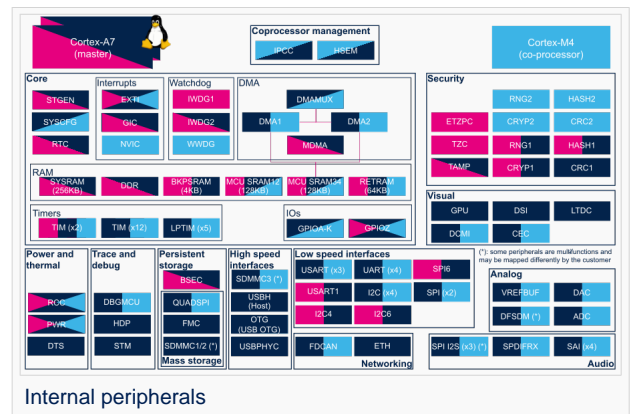
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the [STM32CubeMX](#) tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information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





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 possibilities 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