



TIM internal peripheral

TIM internal peripheral



Contents

1. TIM internal peripheral	3
2. ADC internal peripheral	10
3. DAC internal peripheral	17
4. DFSDM internal peripheral	24
5. How to activate HSI and CSI oscillators calibration	31
6. How to assign an internal peripheral to a runtime context	38
7. IIO overview	45
8. OP-TEE overview	52
9. PWM overview	59
10. RCC internal peripheral	66
11. STM32CubeMP1 architecture	73
12. STM32CubeMX	80
13. STM32MP15 resources	87
14. STM32MPU Embedded Software architecture overview	94
15. TF-A overview	101
16. TIM Linux driver	108
17. TIM device tree configuration	115



A quality version of this page, approved on 5 January 2021, was based off this revision.

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	9
5 References	10



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

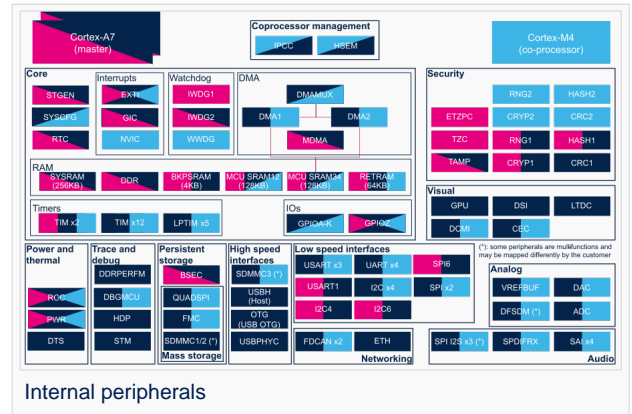
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 16.11.2021 - 14:47 / Revision: 16.11.2021 - 14:47

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	16
5 References	17



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

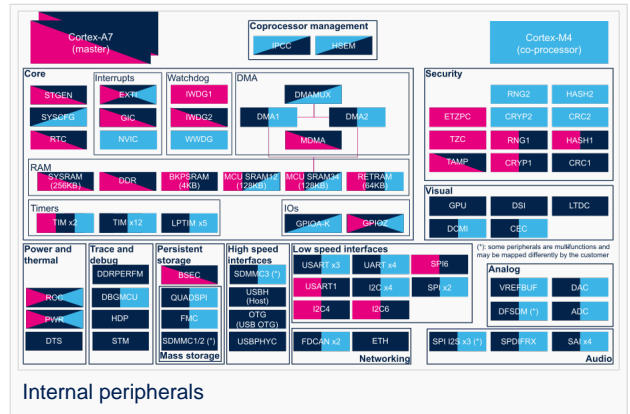
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 12.02.2020 - 16:42 / Revision: 12.02.2020 - 16:41

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	23
5 References	24



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

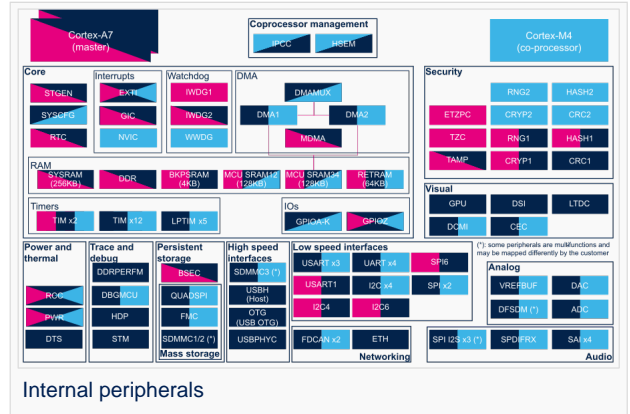
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 12.02.2020 - 16:42 / Revision: 12.02.2020 - 16:41

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	30
5 References	31



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

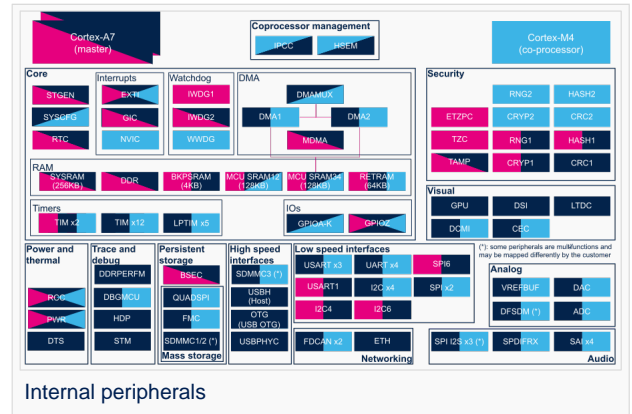
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 03.10.2019 - 12:48 / Revision: 03.10.2019 - 12:47

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	37
5 References	38



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

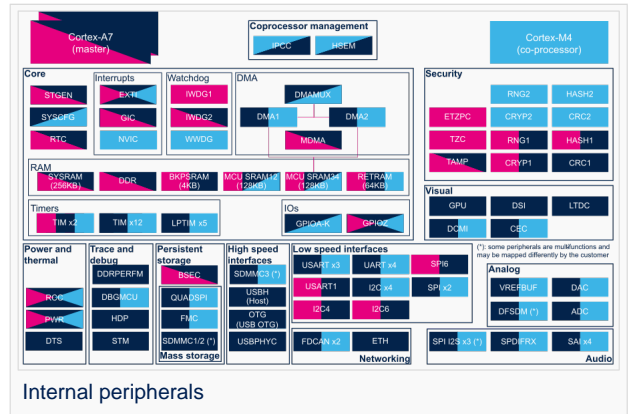
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

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

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	44
5 References	45



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

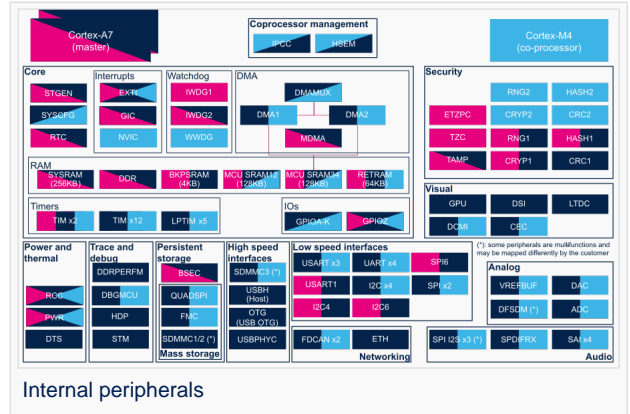
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

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

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	51
5 References	52



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels.

Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)

- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

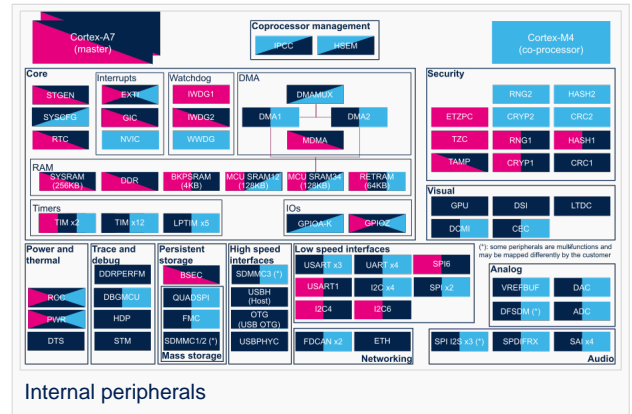
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 13.05.2020 - 08:56 / Revision: 13.05.2020 - 08:54

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	58
5 References	59



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels.

Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)

- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

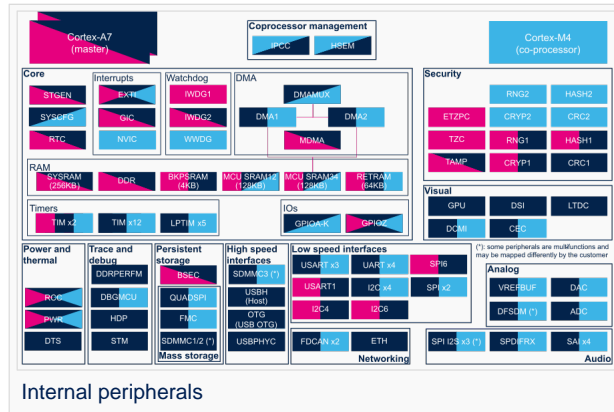
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 18.02.2021 - 13:27 / Revision: 18.02.2021 - 10:37

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	65
5 References	66



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels.

Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)

- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

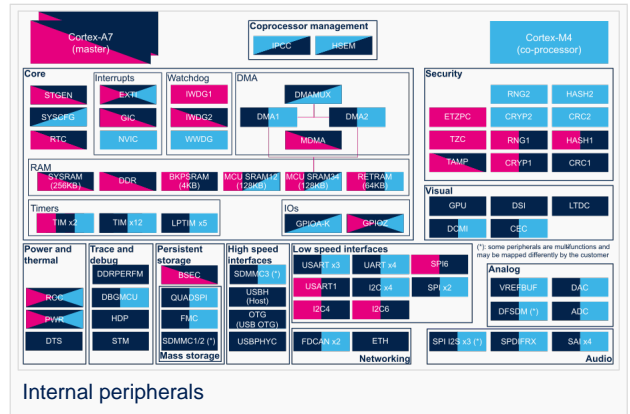
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

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

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	72
5 References	73



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

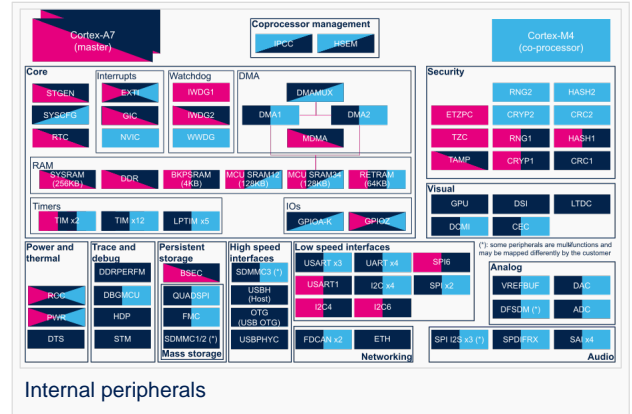
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

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

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	79
5 References	80



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels.

Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)

- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

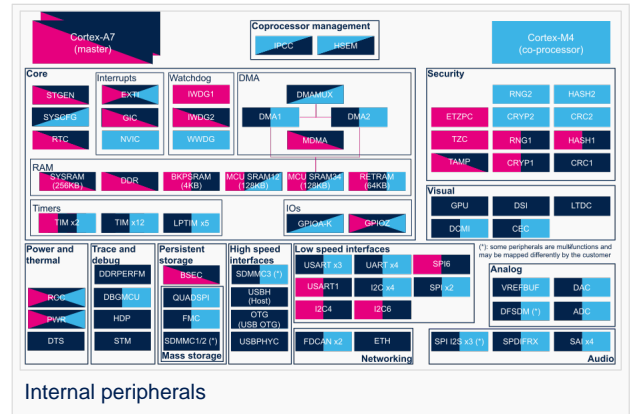
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

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

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations.
We apologize for this inconvenient.

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	86
5 References	87



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

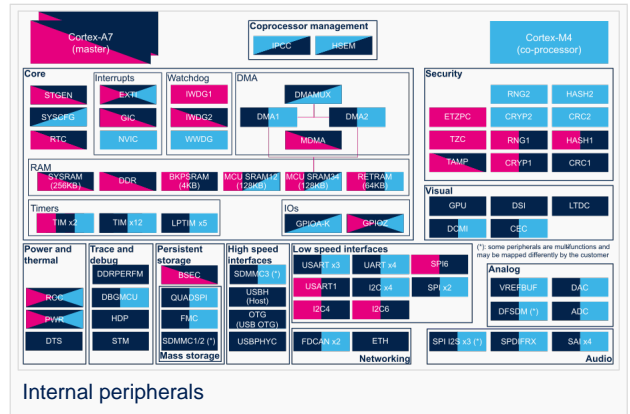
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 17.11.2021 - 16:41 / Revision: 17.11.2021 - 10:47

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	93
5 References	94



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

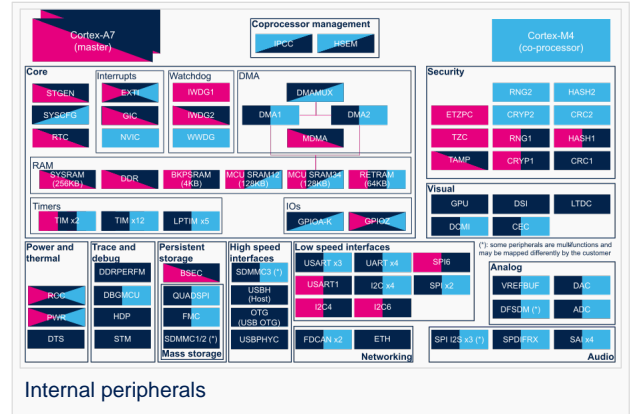
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

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

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	100
5 References	101



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

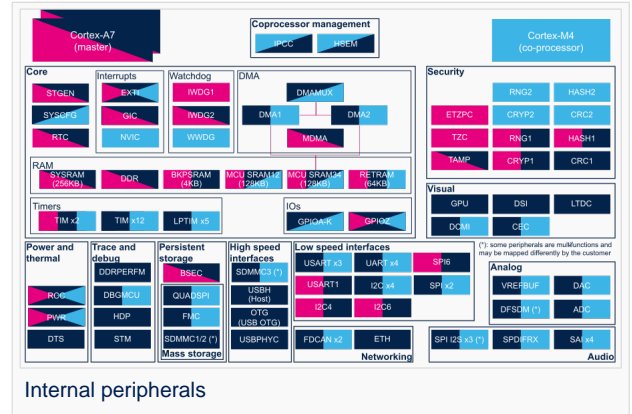
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 22.04.2021 - 11:23 / Revision: 09.04.2021 - 13:17

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	107
5 References	108



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or **TIM15** can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm[®] Cortex[®]-A7 non-secure to be controlled in Linux[®] by the PWM and/or the IIO frameworks.

or

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per **TIM group** corresponding to **APB1** and **APB2** buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to [TIM device tree configuration](#) and [TIM Linux driver](#) articles.

3.2.4 Peripheral assignment

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

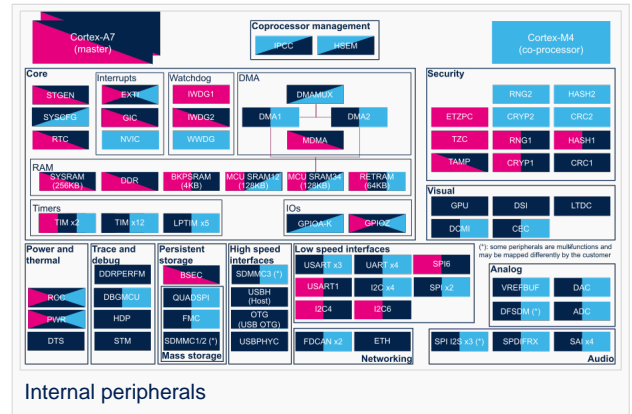
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 26.03.2021 - 15:56 / Revision: 18.03.2021 - 13:55

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations. We apologize for this inconvenient.

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	114
5 References	115



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

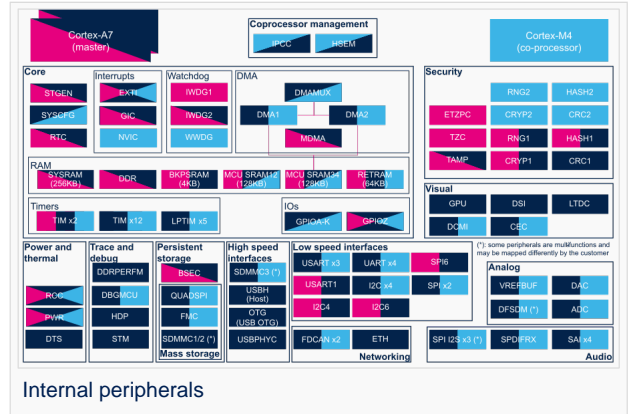
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note

Stable: 26.03.2021 - 16:11 / Revision: 26.03.2021 - 16:01

Warning

This section has not been yet updated/tested for/with the STM32 MPU ecosystem-v3 flow. Information provided in this section may not be not fully applicable for v3 and you may need to proceed to some adaptations.
We apologize for this inconvenient.

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	121
5 References	122



1 Article purpose

The purpose of this article is to

- briefly introduce the **TIM** 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 TIM peripheral



2 Peripheral overview

The TIM peripheral is a multi-channel timer unit, available in various configurations, depending on the instance used. There are basically following categories: advanced-control timers, general-purpose timers and basic timers.

The TIM can provide: PWM with complementary output and dead-time insertion, break detection, input capture^[1], quadrature encoder^[2] interface (typically used for rotary encoders), trigger source for other internal peripherals like: ADC^[3], DAC^[4], DFSDM^[5].

2.1 Features

The **TIM** peripheral is available in different configurations, depending on the selected instance :

- TIM1 and TIM8 are advanced-control timers, with 6 independent channels.
- TIM2, TIM3, TIM4 and TIM5 are general-purpose timers, with 4 independent channels.
- TIM12, TIM13 and TIM14 are general-purpose timers, with 2 (TIM12) or 1 (TIM13 and TIM14) independent channels.
- TIM15, TIM16 and TIM17 are also general-purpose timers, with 2 (TIM15) or 1 (TIM16 and TIM17) independent channels. Compare to TIM12, TIM13 and TIM14, this configuration brings some features that are very useful for motor control (like break function, DMA burst mode control, complementary output with dead-time insertion, ...)
- TIM6 and TIM7 are basic timers

Refer to [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The TIM is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The TIM is not used at boot time.

3.2 Runtime

3.2.1 Overview

TIM12 and/or TIM15 can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in the secure monitor (TF-A or OP-TEE), to perform HSI and CSI calibrations^[6] in RCC.

All TIM instances can be allocated to:

- the Arm® Cortex®-A7 non-secure to be controlled in Linux® by the PWM and/or the IIO frameworks.

or

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by TIM HAL driver

Note that RCC^[7] owns one prescaler per TIM group corresponding to APB1 and APB2 buses: TIMG1PRE and TIMG2PRE, respectively. The allocation to Cortex-A7 or the Cortex-M4 should ideally be done on a per group basis to get independent clocking setup on each side, this is why the TIM instances groups are shown in the summary table below (#Peripheral assignment).

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core/Timers	TIM	TF-A TIM driver OP-TEE TIM driver	Linux PWM framework Linux IIO framework	STM32Cube TIM 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 Linux kernel configuration, please refer to TIM device tree configuration and TIM Linux driver articles.

3.2.4 Peripheral assignment

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

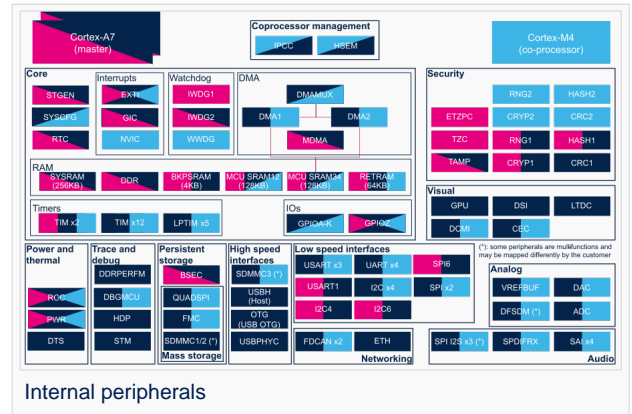
- means that the peripheral can be assigned () to the given runtime context.



- 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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Core/Timers	TIM	TIM1 (APB2 group)		Assignment (single choice)
		TIM2 (APB1 group)		Assignment (single choice)
		TIM3 (APB1 group)		Assignment (single choice)
		TIM4 (APB1 group)		Assignment (single choice)
		TIM5 (APB1 group)		Assignment (single choice)
		TIM6 (APB1 group)		Assignment (single choice)
		TIM7 (APB1 group)		Assignment (single choice)
		TIM8 (APB2 group)		Assignment (single choice)



Domain	Peripherals	Runtime allocation				Comment
		TIM12 (APB1 group)				Assignment (single choice)
		TIM13 (APB1 group)				Assignment (single choice)
		TIM14 (APB1 group)				Assignment (single choice)
		TIM15 (APB2 group)				Assignment (single choice)
		TIM16 (APB2 group)				Assignment (single choice)
		TIM17 (APB2 group)				Assignment (single choice)



4 How to go further

STM32 cross-series timer overview^[8] application note.



5 References

- Input capture
- Quadrature encoder
- ADC internal peripheral
- DAC internal peripheral
- DFSDM internal peripheral
- How to activate HSI and CSI oscillators calibration
- RCC internal peripheral
- STM32 cross-series timer overview application note