

Contents

1. FDCAN internal peripheral	2
2. CAN overview	9
3. FDCAN device tree configuration	16
4. How to assign an internal peripheral to a runtime context	23
5. STM32CubeMP1 architecture	30
6. STM32CubeMX	37
7. STM32MP15 resources	44
8. STM32MPU Embedded Software architecture overview	51

FDCAN internal peripheral

Contents

1	Article purpose	3
2	Peripheral overview	4
2.1	Features	4
2.2	Security support	4
3	Peripheral usage and associated software	5
3.1	Boot time	5
3.2	Runtime	5
3.2.1	Overview	5
3.2.2	Software frameworks	5
3.2.3	Peripheral configuration	5
3.2.4	Peripheral assignment	5
4	How to go further	7
5	References	8

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

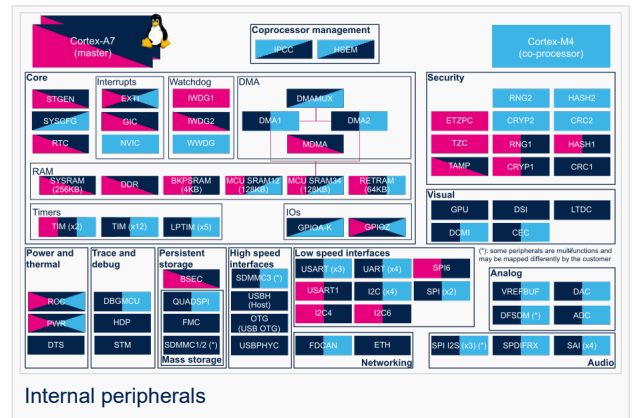
3.2.4 Peripheral assignment

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](#)



Internal peripherals

Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	10
2	Peripheral overview	11
2.1	Features	11
2.2	Security support	11
3	Peripheral usage and associated software	12
3.1	Boot time	12
3.2	Runtime	12
3.2.1	Overview	12
3.2.2	Software frameworks	12
3.2.3	Peripheral configuration	12
3.2.4	Peripheral assignment	12
4	How to go further	14
5	References	15

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

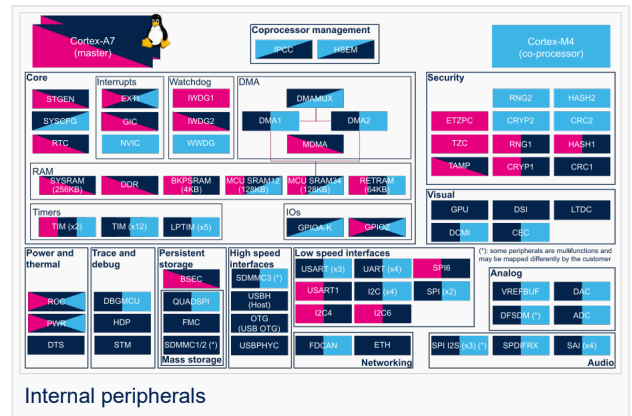
3.2.4 Peripheral assignment

Check boxes illustrate the possible peripheral allocations supported by [STM32 MPU Embedded Software](#):

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

Refer to [How to assign an internal peripheral to a runtime context](#) for more information on how to assign peripherals manually or via STM32CubeMX.

The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possibilities might be described in [STM32MP15 reference manuals](#)



Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	17
2	Peripheral overview	18
2.1	Features	18
2.2	Security support	18
3	Peripheral usage and associated software	19
3.1	Boot time	19
3.2	Runtime	19
3.2.1	Overview	19
3.2.2	Software frameworks	19
3.2.3	Peripheral configuration	19
3.2.4	Peripheral assignment	19
4	How to go further	21
5	References	22

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

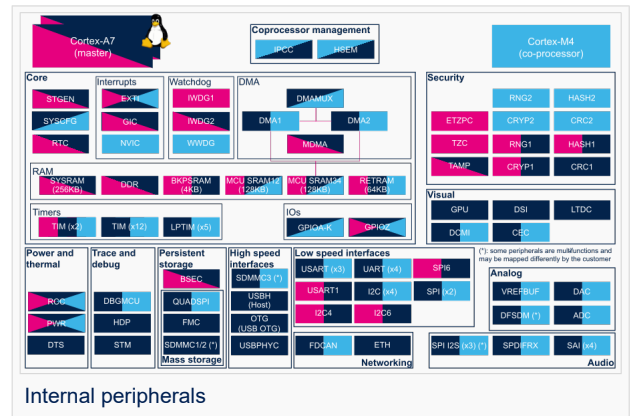
3.2.4 Peripheral assignment

Check boxes illustrate the possible peripheral allocations supported by [STM32 MPU Embedded Software](#):

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

Refer to [How to assign an internal peripheral to a runtime context](#) for more information on how to assign peripherals manually or via STM32CubeMX.

The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possibilities might be described in [STM32MP15 reference manuals](#)



Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	24
2	Peripheral overview	25
2.1	Features	25
2.2	Security support	25
3	Peripheral usage and associated software	26
3.1	Boot time	26
3.2	Runtime	26
3.2.1	Overview	26
3.2.2	Software frameworks	26
3.2.3	Peripheral configuration	26
3.2.4	Peripheral assignment	26
4	How to go further	28
5	References	29

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

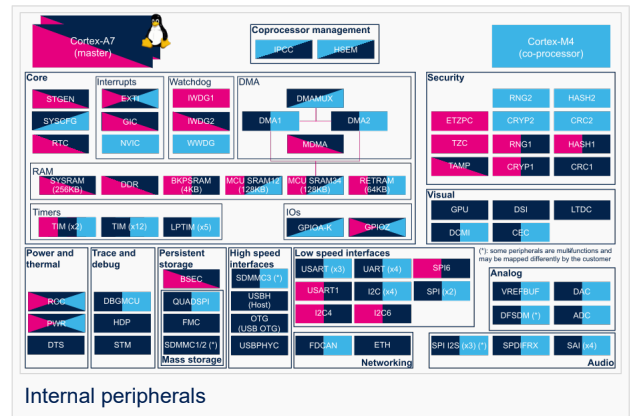
3.2.4 Peripheral assignment

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](#)



Internal peripherals

Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	31
2	Peripheral overview	32
2.1	Features	32
2.2	Security support	32
3	Peripheral usage and associated software	33
3.1	Boot time	33
3.2	Runtime	33
3.2.1	Overview	33
3.2.2	Software frameworks	33
3.2.3	Peripheral configuration	33
3.2.4	Peripheral assignment	33
4	How to go further	35
5	References	36

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

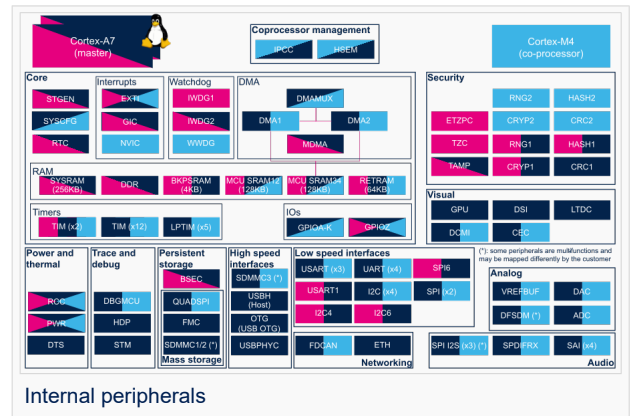
3.2.4 Peripheral assignment

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](#)



Internal peripherals

Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	38
2	Peripheral overview	39
2.1	Features	39
2.2	Security support	39
3	Peripheral usage and associated software	40
3.1	Boot time	40
3.2	Runtime	40
3.2.1	Overview	40
3.2.2	Software frameworks	40
3.2.3	Peripheral configuration	40
3.2.4	Peripheral assignment	40
4	How to go further	42
5	References	43

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

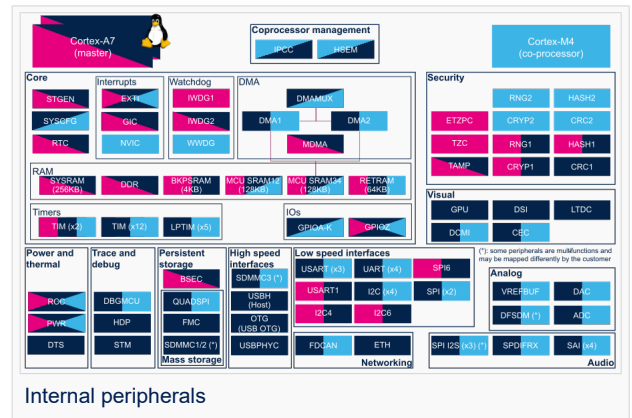
3.2.4 Peripheral assignment

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](#)



Internal peripherals

Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	45
2	Peripheral overview	46
2.1	Features	46
2.2	Security support	46
3	Peripheral usage and associated software	47
3.1	Boot time	47
3.2	Runtime	47
3.2.1	Overview	47
3.2.2	Software frameworks	47
3.2.3	Peripheral configuration	47
3.2.4	Peripheral assignment	47
4	How to go further	49
5	References	50

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

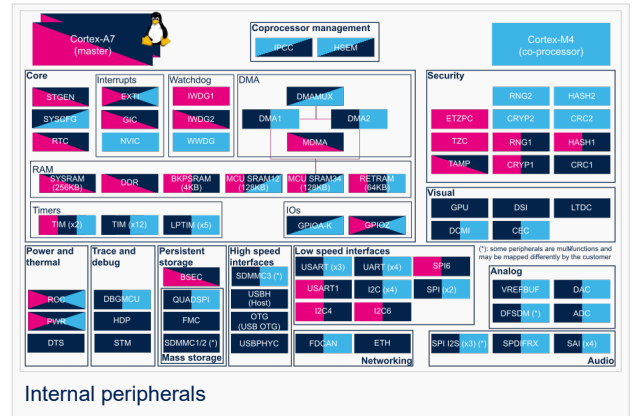
3.2.4 Peripheral assignment

Check boxes illustrate the possible peripheral allocations supported by [STM32 MPU Embedded Software](#):

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

Refer to [How to assign an internal peripheral to a runtime context](#) for more information on how to assign peripherals manually or via STM32CubeMX.

The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possibilities might be described in [STM32MP15 reference manuals](#)



Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

FDCAN internal peripheral

Contents

1	Article purpose	52
2	Peripheral overview	53
2.1	Features	53
2.2	Security support	53
3	Peripheral usage and associated software	54
3.1	Boot time	54
3.2	Runtime	54
3.2.1	Overview	54
3.2.2	Software frameworks	54
3.2.3	Peripheral configuration	54
3.2.4	Peripheral assignment	54
4	How to go further	56
5	References	57

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

FDCAN peripheral handles data communication in a Controller Area Network (CAN) bus system using message-based protocol originally designed for in-vehicle communication. The CAN subsystem consists of two CAN modules (FDCAN1 and FDCAN2), a shared message RAM and an optional clock calibration unit.

2.1 Features

Both FDCAN instances are compliant with classic CAN protocol^[1] and CAN FD^[2] (CAN with Flexible Data-Rate) protocol. In addition, FDCAN1 supports time triggered CAN (TTCAN).

FDCAN1 and FDCAN2 share a dedicated 10 Kbyte CAN SRAM for message transfers.

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

2.2 Security support

FDCAN is a **non secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FDCAN is not used at boot time.

3.2 Runtime

3.2.1 Overview

FDCAN instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the NetDev framework (See [CAN overview](#))
- or
- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by [STM32Cube FDCAN driver](#)

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN		Linux net/can framework	STM32Cube FDCAN 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 can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals) according to the information given in the corresponding software framework article. When the FDCAN peripheral is assigned to the Linux®OS, it is configured through the device tree according to the information given in the [FDCAN device tree configuration](#) article.

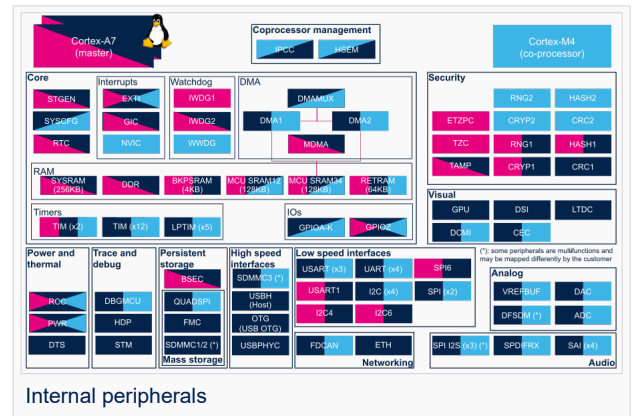
3.2.4 Peripheral assignment

Check boxes illustrate the possible peripheral allocations supported by [STM32 MPU Embedded Software](#):

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

Refer to [How to assign an internal peripheral to a runtime context](#) for more information on how to assign peripherals manually or via STM32CubeMX.

The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possibilities might be described in [STM32MP15 reference manuals](#)



Domain	Peripheral	Runtime allocation			Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Networking	FDCAN	FDCAN1		<input type="checkbox"/>	Assignment (single choice)
		FDCAN2		<input type="checkbox"/>	Assignment (single choice)

4 How to go further

Information

Use this paragraph to add more information and introduce other documentation such as Application Notes (AN)

5 References

- [↑ CAN protocol implementations](#), from the CAN in Automation group (CiA)
- [↑ CAN FD - The basic idea](#), from the CAN in Automation group (CiA)

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2023 STMicroelectronics – All rights reserved