



Elektrobit

EB tresos[®] AutoCore Generic 8 FLASH documentation

Module release 7.0.2





Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

<https://www.elektrobit.com/support>

Legal disclaimer

Confidential and proprietary information

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Elektrobit Automotive GmbH.

All brand names, trademarks and registered trademarks are property of their rightful owners and are used only for description.

Copyright 2019, Elektrobit Automotive GmbH.

Table of Contents

1. Overview	6
2. Flash module release notes	7
2.1. Change log	7
2.2. New features	19
2.3. EB-specific enhancements	19
2.4. Deviations	19
2.5. Limitations	19
3. FLASH user's guide	20
3.1. Overview	20
3.2. Background Information	20
3.2.1. Introduction	20
3.2.2. File list	20
3.2.2.1. Common files for all flash drivers	20
3.2.3. CORTEXM S32K14X Flash driver specific files	21
3.2.4. Feature overview	21
3.3. Configuring FLASH	21
3.3.1. Configuration tab view	21
3.3.1.1. Tab General	22
3.3.1.2. Tab FlsSector	23
3.3.1.3. Tab FlashMemoryAreasProtection	24
3.4. FLASH integration notes	24
3.4.1. Layer header files	25
3.4.2. Layer initialization	25
3.4.3. Flash driver code copy in RAM	25
3.4.3.1. Flash driver linker settings	26
3.4.4. Flash driver Run in Slicer sync mode	26
3.4.5. Flash driver Run in Blocker sync mode	26
3.4.6. Callback function during flash programming	27
3.4.7. Hardware prerequisite	27
4. Flash module references	28
4.1. Configuration parameters	28
4.1.1. CommonPublishedInformation	28
4.1.2. General	31
4.1.3. FlsSector	34
4.1.4. PublishedInformation	35
4.2. Application programming interface (API)	35
4.2.1. Type definitions	36
4.2.1.1. tFlashActivateSecurityCheck	36
4.2.1.2. tFlashAddress	36

4.2.1.3. tFlashBoolean	36
4.2.1.4. tFlashData	36
4.2.1.5. tFlashStatus	36
4.2.1.6. tFlsInfo	36
4.2.2. Macro constants	37
4.2.2.1. COSMIC	37
4.2.2.2. FLASH_ACCESS_ERROR	37
4.2.2.3. FLASH_BASE_ADDR	37
4.2.2.4. FLASH_BLANKCHEK_ERROR	37
4.2.2.5. FLASH_BUSY	37
4.2.2.6. FLASH_COMPILER	38
4.2.2.7. FLASH_ERASED_BYTE_VALUE	38
4.2.2.8. FLASH_FALSE	38
4.2.2.9. FLASH_FAR_ACCESS	38
4.2.2.10. FLASH_FAR_POINTER	38
4.2.2.11. FLASH_FUNC_NOT_SUPPORTED	38
4.2.2.12. FLASH_INFO_SIZE	39
4.2.2.13. FLASH_MANAGE_CASHED_ADDRESS	39
4.2.2.14. FLASH_MANAGE_DIFF_PAGE_SIZES	39
4.2.2.15. FLASH_MANAGE_SECURITY_CHECK	39
4.2.2.16. FLASH_MANAGE_VERIFY	39
4.2.2.17. FLASH_MAX_PROTECTED_AREA_NB	39
4.2.2.18. FLASH_MAX_SECTOR	40
4.2.2.19. FLASH_MODE	40
4.2.2.20. FLASH_MODE_BLOCKER_SYNC	40
4.2.2.21. FLASH_MODE_SLICER_SYNC	40
4.2.2.22. FLASH_NO_ERROR	40
4.2.2.23. FLASH_PAGES_SIZE_BYTE	40
4.2.2.24. FLASH_PAGES_SIZE_WORD	41
4.2.2.25. FLASH_PROTCMD	41
4.2.2.26. FLASH_PROT_ERROR	41
4.2.2.27. FLASH_PROT_REPROG_Register	41
4.2.2.28. FLASH_RAM_COPY_ADDR	41
4.2.2.29. FLASH_REPROG_ENBABLE	41
4.2.2.30. FLASH_SECTOR_SIZE	42
4.2.2.31. FLASH_TOTAL_SIZE	42
4.2.2.32. FLASH_TRUE	42
4.2.2.33. FLASH_WRITE_BUFFER	42
4.2.2.34. GHS	42
4.2.2.35. IAR	42
4.2.2.36. Other	43
4.2.3. Objects	43

4.2.3.1. Fls_Info	43
4.2.3.2. astFlashMemoryAreas	43
4.2.4. Functions	43
4.2.4.1. APP_WatchdogRefresh	43
4.2.4.2. FLASH_Erase	43
4.2.4.3. FLASH_EraseSector	44
4.2.4.4. FLASH_GetJobStatus	44
4.2.4.5. FLASH_GetNextSectorAddr	45
4.2.4.6. FLASH_Init	45
4.2.4.7. FLASH_LLD_Erase	45
4.2.4.8. FLASH_LLD_EraseSector	46
4.2.4.9. FLASH_LLD_Init	46
4.2.4.10. FLASH_LLD_WriteBuffer	46
4.2.4.11. FLASH_MainFunction	47
4.2.4.12. FLASH_Read	47
4.2.4.13. FLASH_WriteBuffer	48
4.3. Integration notes	48
4.3.1. Integration requirements	48



1. Overview

Welcome to the EB tresos AutoCore Generic 8 FLASH product release notes and documentation.

This document provides:

- ▶ [Chapter 2, “Flash module release notes”](#): details of changes and new features in the current release
- ▶ [Chapter 3, “FLASH user's guide”](#): concept information and configuration instructions
- ▶ [Chapter 4, “Flash module references”](#): configuration parameters and the application programming interface

2. Flash module release notes

- ▶ Module version: 7.0.2._B256295
- ▶ Supplier: Elektrobit Automotive GmbH

2.1. Change log

This chapter lists the changes between different versions.

Module version 7.0.2

2019-07-19

Module version 7.0.1

2019-06-20

Module version 7.0.0

2019-05-13

Module version 6.2.6

2019-04-24

Module version 6.2.5

2019-03-13

Module version 6.2.4

2019-02-25



Module version 6.2.3

2019-02-11

Module version 6.2.2

2019-02-07

Module version 6.2.1

2019-01-30

Module version 6.2.0

2018-12-19

Module version 6.1.3

2018-10-25

Module version 6.1.2

2018-10-23

Module version 6.1.1

2018-10-18

Module version 6.1.0

2018-10-17



Module version 6.0.9

2018-10-17

Module version 6.0.8

2018-06-29

Module version 6.0.7

2018-06-27

Module version 6.0.6

2018-06-13

Module version 6.0.5

2018-06-11

Module version 6.0.4

2018-06-07

Module version 6.0.3

2018-05-22

Module version 6.0.2

2018-05-18

Module version 6.0.1

2018-05-16

- ▶ * OSCFLASH-512: Created flash driver

Module version 6.0.0

2018-04-17

Module version 5.1.4

2018-04-17

Module version 5.1.3

2018-03-06

Module version 5.1.2

2018-02-21

Module version 5.1.1

2018-02-08

Module version 5.1.0

2018-02-04



Module version 5.0.2

2018-01-30

Module version 5.0.1

2018-01-29

Module version 5.0.0

2018-01-24

Module version 4.4.1

2018-01-14

Module version 4.4.0

2018-01-12

Module version 4.3.1

2017-12-21

Module version 4.3.0

2017-12-19

Module version 4.2.6

2017-12-18



Module version 4.2.5

2017-12-12

Module version 4.2.4

2017-12-11

Module version 4.2.3

2017-12-10

Module version 4.2.1

2017-11-30

Module version 4.2.0

2017-11-27

Module version 4.1.0

2017-11-07

Module version 4.0.1

2017-10-27

Module version 4.0.0

2017-10-27



Module version 3.1.0

2017-09-05

Module version 3.0.3

2017-09-04

- ▶ OSCFLASH-433 : [common/common] * Fix minimal limitation of 'FLASH_Number_Of_Page_Per_Slicer' in Tresos Studio.

Module version 3.0.2

2017-07-18

Module version 3.0.1

2017-07-14

Module version 3.0.0

2017-07-04

Module version 2.6.2

2017-03-20

Module version 2.6.1

2017-03-14



Module version 2.6.0

2017-01-25

Module version 2.5.6

2017-01-11

Module version 2.5.5

2016-12-22

Module version 2.5.4

2016-12-02

Module version 2.5.3

2016-11-24

Module version 2.5.2

2016-11-13

Module version 2.5.1

2016-10-13

- ▶ OSCFLASH-341 : Depending of the target, public APIs of unimplemented features have been removed.

Module version 2.5.0

2016-09-14



Module version 2.4.9

2016-09-01

Module version 2.4.8

2016-08-15

Module version 2.4.7

2016-07-22

Module version 2.4.6

2016-06-24

Module version 2.4.5

2016-05-27

Module version 2.4.4

2016-05-24

Module version 2.4.3

2016-05-12

Module version 2.4.1

2016-04-20



Module version 2.4.0

2016-04-08

Module version 2.3.1

2016-03-23

Module version 2.3.0

2016-03-22

- ▶ OSCFLASH-242: * Modification FLASH_GetAssociatedSectors to leave last sector during erasing

Module version 2.2.1

2016-02-25

Module version 2.2.0

2016-02-24

- ▶ OSCFLASH-239: * FLASH_Read unused variables removed

Module version 2.1.0

2016-01-14

- ▶ OSCFLASH-231: * FLASH_Read API reworked for more efficient code

Module version 2.0.1

2015-11-19

Module version 2.0.0

2015-11-17

- ▶ OSCFLASH-187: * Wrong reference of JDD module removed

Module version 1.9.0

2015-09-28

- ▶ OSCFLASH-133: * Update interface with AUTOSAR FIs and Resource modules.

Module version 1.8.8

2015-09-01

Module version 1.8.7

2015-08-26

Module version 1.8.6

2015-08-10

Module version 1.8.5

2015-07-23

- ▶ OSCFLASH-126: Code size optimization
- ▶ OSCFLASH-114: Adapt TU to test target independant implementation (FLASH_Prg.c) on WINDOWS

Module version 1.8.4

2015-06-29

- ▶ OSCFLASH-127: OsekCore Flash wrapper for AUTOSAR FIs driver release



Module version 1.8.3

2015-05-30

- ▶ OSCFLASH-118: Create OsekCore Flash wrapper for AUTOSAR Fls driver

Module version 1.8.2

2015-04-30

Module version 1.8.1

2014-10-14

Module version 1.8.0

2014-09-26

Module version 1.7.0

2014-08-06

Module version 1.6.2

2014-07-10

Module version 1.6.1

2014-05-23

Module version 1.6.0

2014-02-14

Module version 1.5.1

2013-10-24

Module version 1.5.0

2013-10-11

2.2. New features

- ▶ No new features have been added since the last release.

2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

2.4. Deviations

This module is not part of the AUTOSAR specification.

2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- ▶ For this module no limitations are known.

3. FLASH user's guide

3.1. Overview

This user's guide describes the `FLASH` module. From this user's guide you will learn about the basic functionality of the `FLASH`. You will also learn which related modules are necessary to configure the `FLASH` module. The `FLASH` module reference provides further information on configuring the `FLASH` itself.

Note that this user's guide is intended for readers who have good knowledge of OsekCore and about the purpose of the `FLASH`. The information provided here should help you to integrate the `FLASH` in your OsekCore project.

- ▶ Section [Section 3.2, "Background Information"](#) provides an overview of the basic functionality of the `FLASH`.
- ▶ Section [Section 3.3, "Configuring FLASH"](#) provides information on related modules that are needed in order to configure the `FLASH`.

3.2. Background Information

3.2.1. Introduction

The `FLASH` Driver layer performs the following functionalities in Blocker sync or Slicer sync mode:

- ▶ Erase multiple or unique flash memory sector in one API call
- ▶ Write a data buffer in flash memory
- ▶ Read a data buffer in flash memory
- ▶ Verify if data have been written in flash memory

3.2.2. File list

The `FLASH` module is composed of the following files:

3.2.2.1. Common files for all flash drivers

- ▶ `FLASH_Cfg.h` : configuration parameters

- ▶ FLASH_Cfg.c : configuration parameters
- ▶ FLASH_Prg.c : target independent layer core implementation
- ▶ FLASH_Pub.h : public prototypes and public types for target independent layer
- ▶ FLASH_LLD.c : target dependent layer core implementation
- ▶ FLASH_LLD.h : public prototypes and public types for target dependent layer
- ▶ FLASH_Cbk.h : prototypes for the callbacks used into the layer core (to be implemented into an upper layer)
- ▶ FLASH_Hw.h : registers address of the microcontroller

3.2.3. CORTEXM S32K14X Flash driver specific files

3.2.4. Feature overview

The `FLASH` module includes features that can be used by configuration. The available features are:

- ▶ Memory areas protection
- ▶ Write Page management
- ▶ Flash driver execution from RAM
- ▶ Choose flash run in Blocker sync mode or Slicer sync mode

3.3. Configuring FLASH

The FLASH layer is designed to be configured using ELEKTROBIT tresos studio software. This step adapts the configuration files from tresos studio module settings.

3.3.1. Configuration tab view

At least one tab, which contains several mandatory information, needs to be configured in the Flash module. Optionally the configuration of other tabs is needed to provide information concerning Flash sectors.

This Section contains the following configuration parameters (not all parameters available for all derivatives):

3.3.1.1. Tab General

This tab contains the following configuration parameters (not all parameters available for all derivatives):

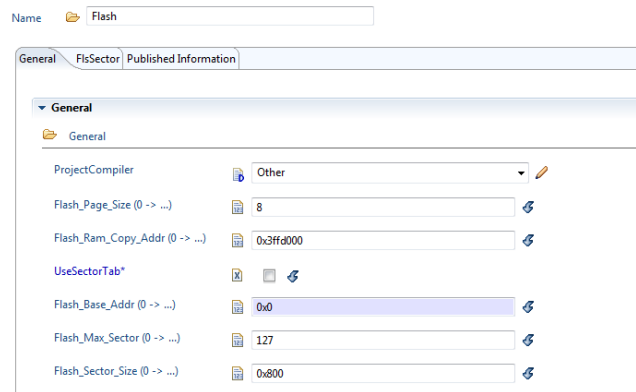


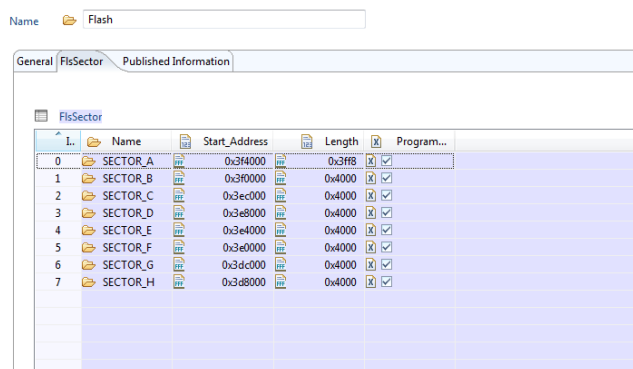
Figure 3.1. FLASH configuration tab "General"

- ▶ **ProjectCompiler** : This parameter contains the compiler used to build the Flash plugin. the define FLASH_COMPILER will be set according the configured value.
- ▶ **Flash_Page_Size** : This parameter contains page size of the Flash memory in bytes. In other word, the minimum number of data bytes that can be written. The define FLASH_PAGES_SIZE_BYTE will be set according the configured value.
- ▶ **Flash_Ram_Copy_Addr** : This parameter contains the address where the Flash driver shall copy the RAM section(s). The define FLASH_RAM_COPY_ADDR will be set according the configured value.
- ▶ **ErasedByteValue** : This parameter indicated which flash technology is used. If the erased state of the Flash cell is logical 0, ErasedByteValue = 0x00. If the erased state of the Flash cell is logical 1, ErasedByteValue = 0xFF.
- ▶ **UseSectorTab** : If this check-box is set, this will activate the second configuration tab (FlsSector), where we provide Flash sector information (such as address or length). If this check-box is not set, the second configuration tab (FlsSector) is deactivated, and the parameters below need to be configured. These parameters will then apply for all identical sectors.
- ▶ **Flash_Base_Addr** : This parameter contains the memory address of the first Flash sector. The define FLASH_BASE_ADDR will be set according the configured value.
- ▶ **Flash_Max_Sector** : This parameter contains the number of Flash sectors. The define FLASH_MAX_SECTOR will be set according the configured value.
- ▶ **Flash_Sector_Size** : This parameter contains the size in bytes of a Flash sector. The define FLASH_SECTOR_SIZE will be set according the configured value.
- ▶ **ActivateSecurityCheck** : If this check-box is set, this will activate the third configuration tab (FlashMemoryAreasProtection), where we provide protected memory area information (such as address or length). If this check-box is not set, the third configuration tab (FlashMemoryAreasProtection) is deactivated.

- ▶ **DifferentProgramAndDataFlashWritePage** : If this check-box is set, the Flash driver will calculate the page size needed for write operation according the Flash type (Data or Program Flash).
- ▶ **CashedAddressConversion** : If this check-box is set, the Flash driver will convert non cashed address to cashed address for write and erase operations.
- ▶ **Flash_Mode** :This parameter make the user to choose the flash is working under Slicer_sync or Blocker_sync mode. Slicer mode is set when you want you real operation is executed in main_function but not the function you called which will return immediatly. Blocer mode is the way you want your operation finished within the called function;
- ▶ **FLASH_Number_Of_Page_Per_Slicer** :This parameter is to indicate how many pages needed to be write in one time in Slicer_sync mode.And this parameter only editable in Slicer_sync mode.
- ▶ **Operation_Callback**:If user want to do some task during writing or eraing operation he can configure Operation_Callback to the function will be called if nothing need to be done just set to NULL_CALLBACK.

3.3.1.2. Tab FlsSector

This tab is only available when the check-box General/UseSectorTab is checked, which means that all Flash sectors are not identical and the characteristic needs to be configured for each sector. This tab contains the following configuration parameters, which are stored in the data structure tFlsInfo for each Flash sector.



I.	Name	Start_Address	Length	Program...
0	SECTOR_A	0x3f4000	0x3ff8	<input checked="" type="checkbox"/>
1	SECTOR_B	0x3f0000	0x4000	<input checked="" type="checkbox"/>
2	SECTOR_C	0x3ec000	0x4000	<input checked="" type="checkbox"/>
3	SECTOR_D	0x3e8000	0x4000	<input checked="" type="checkbox"/>
4	SECTOR_E	0x3e4000	0x4000	<input checked="" type="checkbox"/>
5	SECTOR_F	0x3e0000	0x4000	<input checked="" type="checkbox"/>
6	SECTOR_G	0x3dc000	0x4000	<input checked="" type="checkbox"/>
7	SECTOR_H	0x3d8000	0x4000	<input checked="" type="checkbox"/>

Figure 3.2. FLASH configuration tab "FlsSector"

The following configuration parameters needs to be provided for each Flash sector:

- ▶ **Name** : Name of the sector. The configured named is not generated in the source files but allow the user to identify quickly each sector.
- ▶ **Start_Address** : This parameter contains the start address of the Flash sector. The variable tFlsInfo.Start_Address will be set according the configured value.
- ▶ **Length** : This parameter contains the Flash sector size. The variable tFlsInfo.Length will be set according the configured value.

- ▶ **Programmable** : This check-box is checked if the sector can be reprogrammed. In the other case, check-box has to be leave unchecked. The variable `tFIsInfo.Programmable` will be set according the configured value.

3.3.1.3. Tab FlashMemoryAreasProtection

This tab is only available when the check-box General/ActivateSecurityCheck is checked. It shall list the different memory areas that contain sensitive information (passwords, vector tables, etc.) and which shall be protected against erase and write.

Flash

Name

General **FlashMemoryAreasProtection** Published Information

FlashMemoryAreasProtection

Index	Name	Start_Address	Length
0	FlashMemoryAreasProtection_0	0x0	0x1000
1	FlashMemoryAreasProtection_4	0xff	0x1000
2	FlashMemoryAreasProtection_5	0x1fff	0x1000
3	FlashMemoryAreasProtection_1	0x2fff	0x1000
4	FlashMemoryAreasProtection_2	0x3fff	0x1000
5	FlashMemoryAreasProtection_3	0x4fff	0x1000
6	FlashMemoryAreasProtection_6	0x5fff	0x1000
7	FlashMemoryAreasProtection_7	0x6fff	0x1000
8	FlashMemoryAreasProtection_8	0x7fff	0x1000
9	FlashMemoryAreasProtection_9	0x8fff	0x1000

Figure 3.3. FLASH configuration tab "FlashMemoryAreasProtection"

The following configuration parameters needs to be provided for each protected memory area:

- ▶ **Name** : Name of the protected memory area. The configured named is not generated in the source files but allow the user to identify quickly each memory area.
- ▶ **Start_Address** : This parameter contains the start address of the protected memory area. The variable `tFlashActivateSecurityCheck.Start_Address` will be set according the configured value.
- ▶ **Length** : This parameter contains the Flash sector size. The variable `tFlashActivateSecurityCheck.Length` will be set according the configured value.

3.4. FLASH integration notes

These notes are intended to provide a guide for integrating the `FLASH` module into your project. The points addressed are specific to issues that may arise due to limitations in the `FLASH` or in the related modules mentioned above. Further general integration information is provided in the OsekCore documentation.

The following requirements are mandatory to ensure the correct integration of the module into the complete environment.

3.4.1. Layer header files

The `FLASH` layer is designed to work in the ELEKTROBIT OsekCore stack, which as a unique library called `EB_Prj.h`.

The header file call order shall respect the following:

Id:	OSC-INTMAN-FLASH-0023-1	
Version:	1	
Description:	Header file call order shall respect this order: <code>FLASH_Types.h</code> <code>FLASH_Pub.h</code> <code>FLASH_Cfg.h</code> <code>FLASH_Hw.h</code> <code>FLASH_Cbk.h</code>	
Provides coverage to: (Id-Version Variants)	-	OsekCore

3.4.2. Layer initialization

To make the `FLASH` layer fully operational, it needs to be initialized when starting the program (supply of the microcontroller). The `FLASH` layer has its own initialization function that requires no specific parameter.

The prototype of this function is `FLASH_Init()`.

Id:	OSC-INTMAN-FLASH-0001-1	
Version:	1	
Description:	The customer shall call <code>FLASH_Init</code> function at program start. This init API must be called only at program start and must be first API called of the layer.	
Provides coverage to: (Id-Version Variants)	-	OsekCore

3.4.3. Flash driver code copy in RAM

In order to copy Flash driver code and execute it in RAM, the user has to ensure the following:

- ▶ Flash driver need to know the RAM start address to copy at least the following funtions in `FLASH_RAM.c`.
`FLASH_RAM_WriteRoutine` and `FLASH_RAM_EraseRoutine`.
- ▶ Use linker settings to configure the sections containing the code to be copied in RAM.

See chapter [Section 3.4.3.1, “Flash driver linker settings”](#) for parameter details.

3.4.3.1. Flash driver linker settings

This chapter describes the linker settings needed to copy and execute Flash driver code in RAM

3.4.4. Flash driver Run in Slicer sync mode

To make the FLASH Run in Slicer sync mode. The FLASH Have two mode Blocker sync and Slicer sync. In Slicer sync mode user must use additional FLASH_MainFunction() to process current task after call FLASH_Erase() or FLASH_WriteBuffer(). Because in Slicer sync mode the write and erase functions will return immediately after call FLASH_Erase() or FLASH_WriteBuffer() with a status FLASH_BUSY

The prototypes of the additional functions are tFlashStatus FLASH_MainFunction(void) and tFlashStatus FLASH_GetJobStatus(void).

Id:	OSC-INTMAN-FLASH-0024-1	
Version:	1	
Description:	FLASH_GetJobStatus() has three status: FLASH_BUSY:there still remain job of write or erase to process. FLASH_NO_ERROR:task finished FLASH_ACCESS_ERROR:some error occur.	
Provides coverage to: (Id-Version Variants)	-	OsekCore

3.4.5. Flash driver Run in Blocker sync mode

To make the FLASH Run in Blocker sync mode. The FLASH Have two mode Blocker sync and Slicer sync. When in Blocker sync mode the user only need call write and erase functions and check the return value because these functions will wait for the task to finish. For write task you just directly call FLASH_WriteBuffer(); For Erase you need to erase only one section by on section. So call FLASH_GetNextSectorAddr() to get next sector's address.

Id:	OSC-INTMAN-FLASH-0025-1	
Version:	1	
Description:	FLASH_NO_ERROR:task finished or no task at all. FLASH_ACCESS_ERROR:some error occur.	
Provides coverage to: (Id-Version Variants)	-	OsekCore

3.4.6. Callback function during flash programming

In order to do user task during flash programming

Id:	OSC-INTMAN-FLASH-0026-1	
Version:	1	
Description:	If the user want to do some task during the flash programming,he can configure the "Operation_Callback" to be the name of the function. If the flash can not be accessed during the programming,the function must copy from ROM to RAM. The callback fuction always be called during the loop while to check the flash status.	
Provides coverage to: (Id-Version Variants)	-	OsekCore

3.4.7. Hardware prerequisite

Please check Hardware specification section

4. Flash module references

4.1. Configuration parameters

Containers included		
Container name	Multiplicity	Description
CommonPublishedInformation	1..1	Label: Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
General	1..1	This container contains the general proprieties of the node.
FlsSector	1..n	This contain the description of the sectors."
PublishedInformation	1..1	Label: EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

4.1.1. CommonPublishedInformation

Parameters included	
Parameter name	Multiplicity
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version

Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	0	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	ArMinorVersion	
Label	AUTOSAR Minor Version	
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	0	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	ArPatchVersion	
Label	AUTOSAR Patch Version	
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	0	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	SwMajorVersion	
Label	Software Major Version	
Description	Major version number of the vendor specific implementation of the module.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	7	

Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	SwMinorVersion	
Label	Software Minor Version	
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	0	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	SwPatchVersion	
Label	Software Patch Version	
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	2	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	ModuleId	
Label	Numeric Module ID	
Description	Module ID of this module from Module List	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	0	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	VendorId	
Label	Vendor ID	

Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	1	
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

Parameter Name	Release	
Label	Release Information	
Multiplicity	1..1	
Type	STRING_LABEL	
Default value		
Configuration class	PublishedInformation:	
Origin	Elektrobit Automotive GmbH	

4.1.2. General

Parameters included	
Parameter name	Multiplicity
AsmKeyword	1..1
ProjectCompiler	1..1
FLASH_Ctrl_Address	1..1
UseSectorTab	1..1
FLASH_Page_Size	1..1
Flash_Mode	1..1
FLASH_Number_Of_-Page_Per_Slicer	1..1
Watchdog_care	1..1
Operation_Callback	1..1

Parameter Name	AsmKeyword
Description	Define syntaxe to be used with asm code(asm / __asm / ..).

Multiplicity	1..1
Type	STRING
Default value	asm
Origin	EB

Parameter Name	ProjectCompiler
Description	Select in the list the compiler of the project. It sets correct keywords for the selected one.
Multiplicity	1..1
Type	STRING
Default value	Other
Range	COSMIC
	GHS
	WINDRIVER
	HIGHTEC
	Other
Origin	EB

Parameter Name	FLASH_Ctrl_Address
Description	Define the base address of the Flash controller.
Multiplicity	1..1
Type	INTEGER
Default value	0xC3F88000
Range	>=0
	<=0xFFFFFFFF
Origin	EB

Parameter Name	UseSectorTab
Description	Enable the FlsSector tab, used to map the Flash memory. Mandatory if the flash mapping is not linear.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Origin	EB

Parameter Name	FLASH_Page_Size
Description	Define the base size of a flash page.
Multiplicity	1..1
Type	INTEGER
Default value	8
Range	≥ 0
Origin	EB

Parameter Name	Flash_Mode
Description	<ul style="list-style-type: none"> ► Slicer_Sync: Write and erase operations key parameters are stored within called function which only return FLASH_BUSY, then call FLASH_MainFunction to execute remaining task one slice by one slice and call finally call FLASH_GetJobStatus to check the status of the task; ► Blocker_Sync: Write and erase operations are done within called function;
Multiplicity	1..1
Type	STRING
Default value	Blocker_Sync
Range	Slicer_Sync Blocker_Sync
Origin	EB

Parameter Name	FLASH_Number_Of_Page_Per_Slicer
Description	<p>In Slicer mode, the number of pages written at each FLASH_MainFunction call. This value should be set to biggest within following limitation: 1. considering the external time constraints and write duration of 1 page Example: If 1 page write operation duration is 2ms. While constraints is 100ms which means a status frame should be send within 100ms by bootloader. FLASH_Number_Of_Page_Per_Slicer should set blow to: $100/2 = 50$ and as close to 50 as it can. 2. Considering Stack size because $(FLASH_NUMBER_OF_PAGE_PER_SLICER * FLASH_PAGES_SIZE_BYTE)$ will be used in FLASH_MainFunction as a size of a Array.</p>
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	≥ 1 $\leq 0x80$

Origin	EB
---------------	----

Parameter Name	Watchdog_care
Description	If this option is selected, the FLASH driver calls API to disable the watchdog before memory operation, then API to re-enable the watchdog after the operation. The called APIs are: * void Watchdog_WdgDeactivate(void) * void Watchdog_WdgActivate(void)
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Origin	EB

Parameter Name	Operation_Callback
Description	Pointer to function so that some function can be called in FLASH_LLD_WaitOperationEnding. If it is not necessary to provide the Callback service, disable it by a NULL_CALLBACK field value (default value).
Multiplicity	1..1
Type	STRING
Default value	NULL_CALLBACK
Origin	EB

4.1.3. FlsSector

Parameters included	
Parameter name	Multiplicity
Start_Address	1..1
Length	1..1
Programmable	1..1

Parameter Name	Start_Address
Description	Contains the Flash Start_Address in hexadecimal
Multiplicity	1..1
Type	INTEGER
Origin	EB

Parameter Name	Length
Description	Contains the sector Length in hexadecimal
Multiplicity	1..1
Type	INTEGER
Origin	EB

Parameter Name	Programmable
Description	Specify if the sector is programmable or not
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Origin	EB

4.1.4. PublishedInformation

Parameters included	
Parameter name	Multiplicity
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the Flash can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

4.2. Application programming interface (API)

4.2.1. Type definitions

4.2.1.1. tFlashActivateSecurityCheck

Purpose	Structure of protected memory areas containing adress and length information.	
Type	struct	
Members	u32 Start_Address	
	u32 Length	

4.2.1.2. tFlashAddress

Purpose	Define the flash address type.
Type	const far gpage tFlashData *

4.2.1.3. tFlashBoolean

Purpose	Boolean for FLASH layer.
Type	u8

4.2.1.4. tFlashData

Purpose	Define the data flash type.
Type	u32

4.2.1.5. tFlashStatus

Purpose	Status of flash operations of data flash type.
Type	u8

4.2.1.6. tFlsInfo

Purpose	Structure of flash sector containing adress, length and programmability information.
----------------	--------------------------------------------------------------------------------------

Type	struct	
Members	u32 Start_Address	
	u32 Length	
	u16 Number	
	u8 Programmable	

4.2.2. Macro constants

4.2.2.1. COSMIC

Purpose	Define the use of COSMIC compiler.
Value	0U

4.2.2.2. FLASH_ACCESS_ERROR

Purpose	The flash memory cannot be accessed correctly.
Value	0x02U

4.2.2.3. FLASH_BASE_ADDR

Purpose	Define the base adress of the flash memory.
Value	[!"num:inttohex(General/Flash_Base_Addr)"!]U

4.2.2.4. FLASH_BLANKCHEK_ERROR

Purpose	The flash memory is blank so the data have not been written correctly.
Value	0x04U

4.2.2.5. FLASH_BUSY

Purpose	The flash memory is busy.
----------------	---------------------------

Value	0x05U
--------------	-------

4.2.2.6. FLASH_COMPILER

Purpose	Define the compiler used.
Value	COSMIC

4.2.2.7. FLASH_ERASED_BYTE_VALUE

Purpose	Define the the erased state of the Flash cell.
Value	["(General/ErasedByteValue)!"]U

4.2.2.8. FLASH_FALSE

Purpose	FLASH layer FALSE define used for boolean actions in the layer.
Value	0U

4.2.2.9. FLASH_FAR_ACCESS

Purpose	
Value	

4.2.2.10. FLASH_FAR_POINTER

Purpose	Define how the pointers should be consider according the compiler.
Value	*

4.2.2.11. FLASH_FUNC_NOT_SUPPORTED

Purpose	The function is not supported.
----------------	--------------------------------

Value	0x06U
--------------	-------

4.2.2.12. FLASH_INFO_SIZE

Purpose	Define the size of the sector info tab.
Value	["num:dectoint(count(FlsSector/*))"!U

4.2.2.13. FLASH_MANAGE_CASHED_ADDRESS

Purpose	Define if cashed address conversion shall be managed by the Flash driver.
Value	

4.2.2.14. FLASH_MANAGE_DIFF_PAGE_SIZES

Purpose	Define if different page sizes shall be managed by the Flash driver for Program and Data Flash.
Value	

4.2.2.15. FLASH_MANAGE_SECURITY_CHECK

Purpose	Define if memory area protection shall be managed by the Flash driver.
Value	

4.2.2.16. FLASH_MANAGE_VERIFY

Purpose	Define if FLASH_VerifyStateSection interface shall be provided by the Flash driver.
Value	

4.2.2.17. FLASH_MAX_PROTECTED_AREA_NB

Purpose	Define how many sectors are configured.
----------------	-----------------------------------------

Value	["num:dectoint(count(FlashMemoryAreasProtection/*))"!]U
--------------	---------------------------------------------------------

4.2.2.18. FLASH_MAX_SECTOR

Purpose	Define how many sectors are configured.
Value	["num:dectoint(count(FlsSector/*))"!]U
Description	Define the number of memory sectors configured.

4.2.2.19. FLASH_MODE

Purpose	
Value	FLASH_MODE_BLOCKER_SYNC

4.2.2.20. FLASH_MODE_BLOCKER_SYNC

Purpose	
Value	2U

4.2.2.21. FLASH_MODE_SLICER_SYNC

Purpose	
Value	1U

4.2.2.22. FLASH_NO_ERROR

Purpose	FLASH functions status is OK.
Value	0x01U

4.2.2.23. FLASH_PAGES_SIZE_BYTE

Purpose	Define the size in byte of a page in the flash memory.
----------------	--------------------------------------------------------



Value	[!"num:dectoint(General/Flash_Page_Size)"!]U
--------------	----------------------------------------------

4.2.2.24. FLASH_PAGES_SIZE_WORD

Purpose	Define the size in word byte of a page in the flash memory.
Value	(FLASH_PAGES_SIZE_BYTE / 4U)

4.2.2.25. FLASH_PROTCMD

Purpose	Define the address containing the protection register to allow the flash programming.
Value	*(volatile u8*)(0xFF438004)

4.2.2.26. FLASH_PROT_ERROR

Purpose	The flash memory cannot be accessed correctly due to protection error.
Value	0x03U

4.2.2.27. FLASH_PROT_REPROG_Register

Purpose	Macro modifying status and protection registers.
Value	do{ \ FLASH_PROTCMD=0xa5U; \ FLASH_REPROG_ENBABLE = (value); \ FLASH_REPROG_ENBABLE = ~(value); \ FLASH_REPROG_ENBABLE = (value); \ }while(0);

4.2.2.28. FLASH_RAM_COPY_ADDR

Purpose	Define the address in RAM where to copy the Flash data.
Value	[!"num:inttohex(General/Flash_Ram_Copy_Addr)"!]U

4.2.2.29. FLASH_REPROG_ENBABLE

Purpose	Define the address containing the status register to allow the flash programming.
----------------	-----------------------------------------------------------------------------------

Value	*(volatile u8*)(0xFF438000)
--------------	-----------------------------

4.2.2.30. FLASH_SECTOR_SIZE

Purpose	Define the size of every memory sectors.
Value	["num:inttohex(General/Flash_Sector_Size")!] U

4.2.2.31. FLASH_TOTAL_SIZE

Purpose	Define the total size of the flash memory.
Value	["num:inttohex(\$currentConfiguredSize")!] U

4.2.2.32. FLASH_TRUE

Purpose	FLASH layer FALSE define used for boolean actions in the layer.
Value	1U

4.2.2.33. FLASH_WRITE_BUFFER

Purpose	Define the size of the temporary write buffer.
Value	(FLASH_PAGES_SIZE_BYTE)

4.2.2.34. GHS

Purpose	Define the use of GHS compiler.
Value	1U

4.2.2.35. IAR

Purpose	Define the use of IAR compiler.
Value	2U

4.2.2.36. Other

Purpose	Define the use of Other compiler.
Value	3U

4.2.3. Objects

4.2.3.1. Fls_Info

Purpose	
Type	const tFlsInfo

4.2.3.2. astFlashMemoryAreas

Purpose	Array of structure containing all memory area to be protected against erase and write.
Type	const tFlashActivateSecurityCheck

4.2.4. Functions

4.2.4.1. APP_WatchdogRefresh

Purpose	APP Callback called refresh the watchdog when an operation (e.g erasing) take too long.
Synopsis	<pre>void APP_WatchdogRefresh (void);</pre>

4.2.4.2. FLASH_Erase

Purpose	Erase one or several sectors of flash memory.
Synopsis	<pre>tFlashStatus FLASH_Erase (tFlashAddress uAddr , u32 uwLen);</pre>

Parameters (in)	uAddr	Memory address of the first sector to be erased
	uwLen	Length of the data to be erased
Return Value	Result of operation	
	FLASH_ACCESS_ERROR	An error occurred during erasing operations. The data have not been written correctly.
	FLASH_NO_ERROR	No problem, the sector(s) have been correctly erased
Description	This function allows to erase one or several sectors of the flash memory, starting from the address specified in input parameter.	

4.2.4.3. FLASH_EraseSector

Purpose	Erase sector of flash memory.	
Synopsis	<code>tFlashStatus FLASH_EraseSector (tFlashAddress uAddr);</code>	
Parameters (in)	uAddr	Memory address of the sector to be erased
Return Value	Result of operation	
	FLASH_ACCESS_ERROR	An error occurred during erasing operations. The data have not been written correctly.
	FLASH_NO_ERROR	No problem, the sector have been correctly erased
Description	This function allows to erase the sector of the flash memory, corresponding to the address specified in input parameter	

4.2.4.4. FLASH_GetJobStatus

Purpose	Get the status of the current operation.	
Synopsis	<code>tFlashStatus FLASH_GetJobStatus (void);</code>	
Return Value	Status of the ongoing operation	
Description	this function is used to request the status of the ongoing asynchronous operation (Flash erase or write)	

4.2.4.5. FLASH_GetNextSectorAddr

Purpose	Get next sector start address.	
Synopsis	<code>tFlashAddress FLASH_GetNextSectorAddr (tFlashAddress uAddr);</code>	
Parameters (in)	uAddr	Memory address of reference sector
Return Value	Start address of the next sector. In case the sector address given in parameter is the last one the mapping, the function returns last address of the current sector.	
Description	This function is used to get start address of the sector after the one in which address specified in input parameter belongs to.	

4.2.4.6. FLASH_Init

Purpose	Initialize layer.	
Synopsis	<code>void FLASH_Init (void);</code>	
Description	This function initializes the FLASH layer, shall be called only once at ECU startup	

4.2.4.7. FLASH_LLD_Erase

Purpose	Erase one or several sectors of flash memory.	
Synopsis	<code>tFlashStatus FLASH_LLD_Erase (tFlashAddress uAddr , u16 ulStartBlockNo , u16 ulEndBlockNo);</code>	
Parameters (in)	uAddr	Memory address of the first sector to be erased
	ulStartBlockNo	First block to be erased
	ulEndBlockNo	Last block to be erased
Return Value	Result of operation	
	FLASH_LLD_ACCESS_ERROR	An error occurred during erasing operations. The data have not been written correctly.
	FLASH_LLD_NO_ERROR	No problem, the sector(s) have been correctly erased

	FLASH_PROT_ERROR	Data cannot be erased because address is located in a protected memory area
Description	This function allows to erase one or several sectors of the flash memory, starting from the address specified in input parameter.	

4.2.4.8. FLASH_LLD_EraseSector

Purpose	Erase sector of flash memory.	
Synopsis	<code>tFlashStatus FLASH_LLD_EraseSector (u16 ulSectorNo);</code>	
Parameters (in)	ulSectorNo	Sector to erase
Return Value	Result of operation	
	FLASH_LLD_ACCESS_ERROR	An error occurred during erasing operations. The data have not been written correctly.
	FLASH_LLD_NO_ERROR	No problem, the sector have been correctly erased
	FLASH_PROT_ERROR	Data cannot be erased because address is located in a protected memory area
Description	This function allows to erase the sector of the flash memory, corresponding to the address specified in input parameter	

4.2.4.9. FLASH_LLD_Init

Purpose	Initialize layer.
Synopsis	<code>void FLASH_LLD_Init (void);</code>
Description	This function initializes the FLASH layer, shall be called only once at ECU startup

4.2.4.10. FLASH_LLD_WriteBuffer

Purpose	Write buffer of data to flash memory.
Synopsis	<code>tFlashStatus FLASH_LLD_WriteBuffer (tFlashAddress uAddr , u16 uwLen , const u8 FLASH_FAR_POINTER aubData);</code>

Parameters (in)	uAddr	Memory address where the data shall be written
	uwLen	Length of the data to be written
	aubData	Buffer of data to be written
Return Value	Result of operation	
	FLASH_LLD_ACCESS_ERROR	An error occurred during writing operations. The data have not been written correctly.
	FLASH_LLD_NO_ERROR	No problem, the data have been written correctly
	FLASH_PROT_ERROR	Data cannot be written because address is located in a protected memory area
Description	This function allows to write a buffer of data in flash memory at the address specified in input parameter	

4.2.4.11. FLASH_MainFunction

Purpose	Process Asynchronous operation for flash mode.
Synopsis	<pre>void FLASH_MainFunction (void);</pre>
Description	this function is used to process of the ongoing Asynchronous operation (Flash erase or write)
	return value

4.2.4.12. FLASH_Read

Purpose	Read the flash memory.	
Synopsis	<pre>tFlashStatus FLASH_Read (tFlashAddress uAddr , u16 uwLen , u8 * aubData);</pre>	
Parameters (in)	uAddr	Memory address in flash to read
	uwLen	Length of the data to read
	aubData	Buffer where to store the read data
Return Value	Result of operation	
	FLASH_ACCESS_ERROR	An error occurred during check operations.

	FLASH_NO_ERROR	No problem, the memory has been checked correctly
Description	This function allows to read the data located to the address specified in input parameter.	

4.2.4.13. FLASH_WriteBuffer

Purpose	Write buffer of data to flash memory.	
Synopsis	<pre>tFlashStatus FLASH_WriteBuffer (tFlashAddress uAddr , u16 uwLen , const u8 FLASH_FAR_POINTER aubData);</pre>	
Parameters (in)	uAddr	Memory address where the data shall be written
	uwLen	Length of the data to be written
	aubData	Buffer of data to be written
Return Value	Result of operation	
	FLASH_ACCESS_ERROR	An error occurred during writing operations. The data have not been written correctly.
	FLASH_NO_ERROR	No problem, the data have been written correctly
Description	This function allows to write a buffer of data in flash memory at the address specified in input parameter	

4.3. Integration notes

4.3.1. Integration requirements

WARNING



Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Flash module.