SCOPE OF APPLICATION All Project/Engineering	HYUNDAI AutoEver	SHT/SHTS 1 / 31
Responsibility: Classic AUTOSAR Team	Mem_76_Pfls Manual	DOC. NO

Memory Driver User Manual

Document Change Histroy				
Date (YYYY-MM-DD)	Ver.	Editor	Chap	Content
2022-11-29	1.0.0.0	LinhTT36	All	- Initial version
2023-10-31	1.1.0.0	NhanLT1	5, 8	- Support more MCUs

 Edition Date: 2023-10-31	File Name Mem_76_Pfls_UM.pdf	Creation **NhanLT1** 2023-10- 31	Check **YongSeong Jeon** 2023-10- 31	Approval **YongSeong Jeon** 2023-10- 31
Document Management System				

Table of Contents

1 Overview

The document is written based AUTOSAR standard SRS/SWS, and if more detailed functional description is needed when using the module, refer to the reference below

The interpretation of the category related to setting is as follows

- Changeable (C): Items that can be set by users
- Fixed (F): Items that can not changed by users
- NotSupported (N): Not used

2 Reference

Sl. No.	Title	Version
1	AUTOSAR_SWS_MemoryDriver.pdf	4.4.0

3 AUTOSAR System

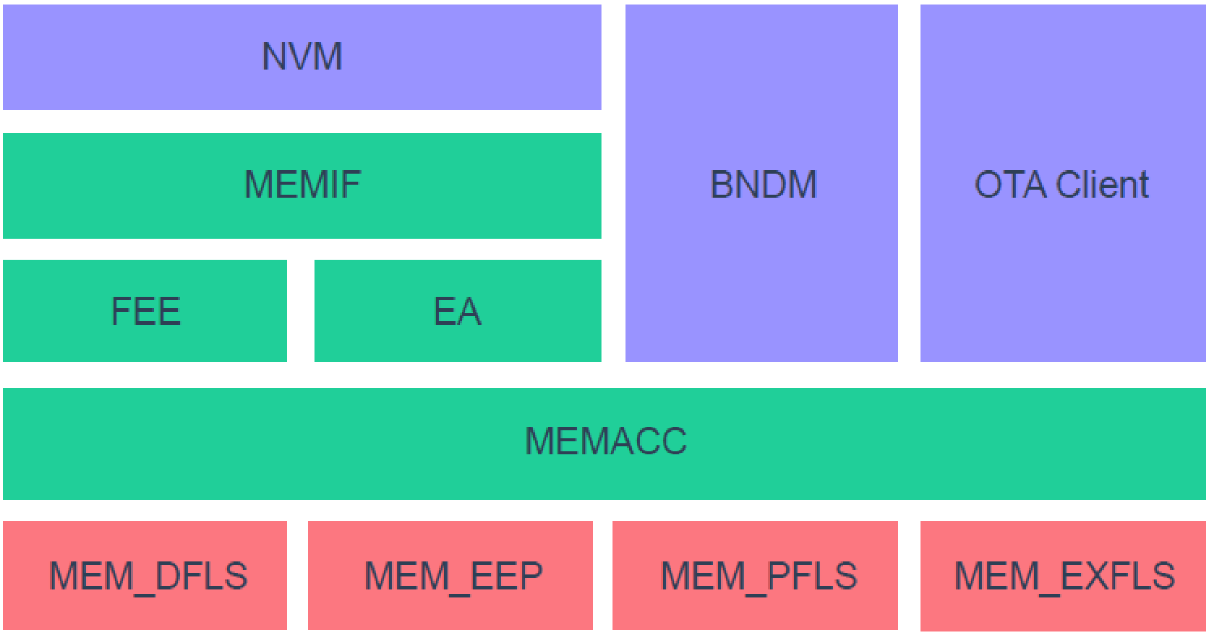
3.1 Memory Module

The MemoryDriver provides the basic services for accessing different kinds of memory devices like reading, writing, erasing and blank checking.

Although flash memory is still the most non-volatile memory technology, the Memory Driver specification considers all relevant memory device technologies like EEPROM, phase change memory (PCM) and ferro electric RAM.

To Harmonize the memory access for the upper layers, the memory driver specification also cover access of RAM. Aside from microcontroller internal devices, the memory device specification can also be applied to external memory device attached via a serial peripheral interface.

In contrast to the Flash and EEPROM Driver specification, the memory driver specification explicitly covers also code memory access to support new use cases like back-group OTA software update which require code memory access.



4 Product Release Notes

4.1 Overview

The purpose of this chapter is to provide information related to Autosar Memory Driver module and describe limitations and specifics of Memory Driver software product release version.

4.2 Scope of the Release

All contents of this document are limited to the following Huyndai AutoEver’s CanSM modules

Module name	AUTOSAR version	Module version
Mem	4.4.0	1.0.0

※ Module version means the Sw version of each module's BswModule Description (Bswmd) file.

4.3 Change Log

4.3.1 Version 1.0.0.0

➢ New version

- New Mem_76_Pfls module development

Cause	New development
Operation effect	None
Setting effect	None
ASW Action	None

4.3.2 Version 1.1.0.0

➢ New version

- Improve Mem_76_Pfls module development

Cause	Support more MCUs
Operation effect	None
Setting effect	None
ASW Action	None

4.4 Module Release Notes

4.4.1 Limitations

- In Aurix 2G(TC39X), ECC bit is read if read as it is after Erase.
  If you need to read 0x00 to check the Erase result, you can read after disabling Safety ECC for the meantime. Alternatively, you can run the Verify Erased Page command attached to the page you want to inspect.
- User have to configure the callout, if the callout is not configured in 'Mem_76_Pfls_User_Callouts' , there will be a compilation error

4.4.2 Deviations

5 Configuration Guide

5.1 MemGeneral

Parameter Name	Value	Category
MemDevErrorDetect [1]	True	C
MemMainFunctionTimePeriod [2]	0.005	C
MemUserIncludeFiles[3]	Det.h	C
MemProcessorSeries[4]	TC39x	C
WdgTriggerCalloutFunction[5]	Mem_76_Pfls_Wdg_Trigger_Callout	C
MemUseNvm[6]	True	C
MemUseRamCode[7]	False	C
MemDisableEccTrap[8]	True	C

1) MemDevErrorDetect

- Select whether to use error notification function

2) MemMainFunctionTimePeriod

- Set the execution cycle (unit: second) of the cycle function

3) MemUserIncludeFiles

- This parameter configure the header file that should be included for user

4) MemProcessorSeries

- This parameter set the MCU name of the current platform that Mem_76_Pfls is integrated

5) WdgTriggerCalloutFunction

- This parameter set the API that shall be called to trigger WatchDog

6) MemUseNvm

- Switches the development error detection and notification on or off

7) MemUseRamCode

- Enable run flash operation in RAM

8) MemDisableEccTrap

- Disable Ecc Trap of Memory Program Flash

5.2 MemPublishedInformation

Parameter Name	Value	Category
MemPublishedInformation[1]	0xFFFFFFFF	F

1) MemPublishedInformation

- The parameter inform user the value of flash memory addres when it is reset to default.

5.3 MemInstance

Parameter Name	Value	Category
MemInstanceId	-	C

This container includes the Mem driver instance specific configuration parameters. Its multiplicity describes the number of Mem driver instances of this Mem driver. There will be one container for each Mem driver instance.

1) MemInstanceId

- This value specifies the unique numeric identifier which is used to reference a Mem driver instance in case multiple devices of the same type shall be addressed by one Mem driver. This value will be assigned to the symbolic name derived of the MemInstance container short name.

5.4 MemInstance-MemSectorBatch

Parameter Name	Value	Category
MemEraseSectorSize(1)	0x2000	F
MemNumberOfSectors(2)	-	C
MemReadPageSize(3)	1	F
MemSpecifiedEraseCycles(4)	0xFFFFFFFF	F
MemStartAddress(5)	-	C
MemAltStartAddress (6)	-	C
MemWritePageSize(7)	8	F

1) MemEraseSectorSize

- Size of a sector in bytes
- A sector is the smallest erasable unit

2) MemNumberOfSectors

- Number of contiguous sectors with identical values for MemSectorSize and MemPageSize.

- If this parameter is configured to be greater than 1, the sectors are grouped in a sector batch.

3) MemReadPageSize

- Size of a read page of this sector in bytes.
- A read page is the smallest readable unit

4) MemSpecifiedEraseCycles

- Number of erase cycles specified for the memory device (usually given in the device data sheet).

5) MemStartAddress

- Physical start address of the sector (batch). In case of a sector batch, the physical start address is the address of the first sector. The physical start address of the other sectors can be
- calculated by the MemSectorSize parameter.

6) MemWritePageSize

- Size of a write page of this sector in bytes.
- A write page is the smallest writeable unit.

7) MemAltStartAddress

- Alternative start address of the sector (batch).
- In case of the sector batch, the alternative start address is the address of the inactive first sector.

5.5 MemInstance-MemSectorBatch-MemBurstSettings

Parameter Name	Value	Category
MemEraseBurstSize(1)	0x200	F
MemReadBurstSize(2)	1	F
MemWriteBurstSize(3)	128	F

Container for burst setting configuration parameters of the Mem driver. A sector burst can be used for improved performance.

1) MemEraseBurstSize

- Size of sector erase burst in bytes. A sector burst can be used for improved performance and is typically (a subset of) a sector batch.
- To make use of the sector erase burst feature, the physical start address of the sector batch must be aligned to the sector erase burst size.

2) MemReadBurstSize

- This value specifies the maximum number of bytes the MemAcc module requests within one Mem read request.

3) MemWriteBurstSize

- Size of page write/program burst in bytes. A sector burst can be used for improved performance and is typically (a subset of) a sector batch.
- To make use of the write burst feature, the physical start address must be aligned to the write burst size.

5.6 MemInstance-MemUserPreCalloutFunction

Parameter Name	Value	Category
UserPreCalloutFunction(1)	Mem_76_Pfls_CacheControlDisable_Callout	C

Container for UserPreCalloutFunction configuration parameters of the Mem driver

1) UserPreCalloutFunction

- This parameter set the API that shall be called before accessing Memory driver code.

5.7 MemInstance-MemUserPostCalloutFunction

Parameter Name	Value	Category
UserPostCalloutFunction(1)	Mem_76_Pfls_CacheControlEnable_Callout	C

Container for UserPostCalloutFunction configuration parameters of the Mem driver

1) UserPostCalloutFunction

- This parameter set the API that shall be called after accessing Memory driver code

6 Application Programming Interface (API)

6.1 Type Definitions

6.1.1 Mem_76_Pfls_AddressType

Name	Mem_76_Pfls_AddressType
Kind	Type
Derived from	Uint32
Description	Physical memory device address type
Available via	Mem_76_Pfls.h

6.1.2 Mem_76_Pfls_ConfigType

Name	Mem_76_Pfls_ConfigType
Kind	Structure
Description	Postbuild configuration structure type
Available via	Mem_76_Pfls.h

6.1.3 Mem_76_Pfls_AddressType

Name	Mem_76_Pfls_DataType
Kind	Type
Derived from	Uinit8
Description	Read data user buffer type
Available via	Mem_76_Pfls.h

6.1.4 Mem_76_Pfls_InstanceIdType

Name	Mem_76_Pfls_MemInstanceIdType
Kind	Type
Derived from	Uinit32
Description	Job end notification function called by Mem_76_Pfls in case the job processing is configured for job end notification.
Available via	Mem_76_Pfls.h

6.1.5 Mem_76_Pfls_JobResultType

Name	Mem_76_Pfls_JobResultType		
Kind	Enumeration		
	MEM_JOB_OK	0x00	The last job has been finished successfully

	MEM_JOB_PENDING	0x01	A job is currently being processed
	MEM_JOB_FAILED	0x02	A job failed for some unspecific reason
	MEM_INCONSISTENT	0x03	The checked page is not blank
	MEM_ECC_UNCORRECTED	0x04	Uncorrectable ECC errors occurred during memory access
	MEM_ECC_CORRECTED	0x05	Correctable ECC errors occurred during memory access
Description	Asynchronous job result type		
Available via	Mem_76_Pfls.h		

6.1.6 Mem_76_Pfls_LengthType

Name	Mem_76_Pfls_LengthType
Kind	Type
Derived from	Uinit32
Description	Physical memory device length type
Available via	Mem_76_Pfls.h

6.1.7 Mem_76_Pfls_HwServiceIdType

Name	Mem_76_Pfls_HwServiceIdType
Kind	Type
Derived from	Uinit32
Description	Hardware specific service request identifier type
Available via	Mem_76_Pfls.h

6.2 Macro Constants

None

6.3 Functions

6.3.1 Synchronous functions

6.3.1.1 Mem_76_Pfls_Init

Function Name	Mem_76_Pfls_Init
Syntax	void Mem_76_Pfls_Init **(**const Mem_ConfigType* configPtr**)**
Service ID	0x01
Sync/Async	Synchronous

Reentrancy	Non Reentrant	
Parameters (In)	ConfigPtr	Pointer to the configuration data structure - since Mem_76_Pfls driver is a precompile module this parameter is typically not used.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	Initialization function - initializes all variables and sets the module state to initialized	
Available via:	Mem_76_Pfls.h	

6.3.1.2 Mem_76_Pfls_DeInit

Function Name	Mem_76_Pfls_DeInit
Syntax	void Mem_76_Pfls_DeInit (void)
Service ID	0x0b
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	De-initialize module. If there is still an access job pending, it is immediately terminated (using hardware cancel operation) and the Mem driver module state is set to unitialized. Therefore, Mem must be re-initialized before it will accept any new job requests after this service is processed.
Available via:	Mem_76_Pfls.h

6.3.1.3 Mem_76_Pfls_GetVersionInfo

Function Name	Mem_76_Pfls_GetVersionInfo
Syntax	void Mem_76_Pfls_GetVersionInfo **(**Std_VersionInfoType***** versionInfoPtr**)**
Service ID	0x02
Sync/Async	Synchronous

Reentrancy	Non Reentrant	
Parameters (In)	versionInfoPtr	Pointer to standard version information structure.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	Service to return the version information of the Mem module.	
Available via:	Mem_76_Pfls.h	

6.3.1.4 Mem_76_Pfls_GetJobResult

Function Name	Mem_76_Pfls_GetJobResult	
Syntax	Mem_76_Pfls_JobResultType Mem_76_Pfls_GetJobResult **(**Mem_InstanceIdType instanceId**)**	
Service ID	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	InstanceId	ID of the related memory driver instance.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Mem_76_Pfls_JobResultType	Most recent job result.
Description	Service to return results of the most recent job.	
Available via:	Mem_76_Pfls.h	

6.3.1.5 Mem_76_Pfls_Suspend

Function Name	Mem_76_Pfls_Suspend	
Syntax	void Mem_76_Pfls_Suspend **(**Mem_InstanceIdType instanceId**)**	
Service ID	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	InstanceId	ID of the related memory driver instance
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	

Description	Suspend active memory operation using hardware mechanism.
Available via:	Mem_76_Pfls.h

6.3.1.6 Mem_76_Pfls_Resume

Function Name	Mem_76_Pfls_Resume	
Syntax	void Mem_76_Pfls_Resume **(**Mem_InstanceIdType instanceId**)**	
Service ID	0x0d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	InstanceId	ID of the related memory driver instance
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	Resume suspended memory operation using hardware mechanism.	
Available via:	Mem_76_Pfls.h	

6.3.1.7 Mem_76_Pfls_PropagateError

Function Name	Mem_76_Pfls_ProgateError	
Syntax	void Mem_76_Pfls_ProgateError **(**Mem_InstanceIdType instanceId**)**	
Service ID	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	InstanceId	ID of the related memory driver instance
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service can be used to report an access error in case the Mem driver cannot provide the access error information - typically for ECC faults. It is called by the system ECC handler to propagate an ECC error to the memory upper layers..	
Available via:	Mem_76_Pfls.h	

6.3.2 Asynchronous Functions

6.3.2.1 Mem_76_Pfls_Read

Service name:	Mem_76_Pfls_Read	
Syntax:	Std_ReturnType Mem_76_Pfls_Read **(** Mem_InstanceIdType instanceId, Mem_AddressType sourceAddress, Mem_DataType* destinationDataPtr, Mem_LengthType length **)**	
Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	instanceId	ID of the related memory driver instance.
	sourceAddress	Physical address to read data from.
	length	Read length in bytes.
Parameters (inout):	None	
Parameters (out):	destinationDataPtr	Destination memory pointer to store the read data.
Return value:	Std_ReturnType	E_OK: The requested job has been accepted by the module. E_NOT_OK: The requested job has not been accepted by the module. E_MEM_SERVICE_NOT_AVAIL: The service function is not implemented.
Description:	Triggers a read job to copy the from the source address into the referenced destination data buffer. The result of this service can be retrieved using the Mem_GetJobResult API. If the read operation was successful, the result of the job is MEM_JOB_OK. If the read operation failed, the result of the job is either MEM_JOB_FAILED in case of a general error or MEM_ECC_CORRECTED/MEM_ECC_UNCORRECTED in case of a correctable/uncorrectable ECC error.	
Available via:	Mem_76_Pfls.h	

6.3.2.2 Mem_76_Pfls_Write

Service name:	Mem_76_Pfls_Write	
Syntax:	Std_ReturnType Mem_76_Pfls_Write **(** Mem_InstanceIdType instanceId, Mem_AddressType sourceAddress, Mem_DataType***** destinationDataPtr, Mem_LengthType length **)**	
Service ID[hex]:	0x06	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	instanceId	ID of the related memory driver instance.
	targetAddress	Physical write address (aligned to page size).
	SourceDataPtr	Source data pointer (aligned to page size).
	length	Write length in bytes (aligned to page size).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The requested job has been accepted by the module. E_NOT_OK: The requested job has not been accepted by the module. E_MEM_SERVICE_NOT_AVAIL: The service function is not implemented.
Description:	Triggers a write job to store the passed data to the provided address area with given address and length. The result of this service can be retrieved using the Mem_GetJobResult API. If the write operation was successful, the job result is MEM_JOB_OK. If there was an issue writing the data, the result is MEM_FAILED.	
Available via:	Mem_76_Pfls.h	

6.3.2.3 Mem_76_Pfls_Erase

Service name:	Mem_76_Pfls_Erase	
Syntax:	Std_ReturnType Mem_76_Pfls_Erase **(** Mem_InstanceIdType instanceId, Mem_AddressType targetAddress, Mem_LengthType length**)**	
Service ID[hex]:	0x07	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	instanceId	ID of the related memory driver instance.
	targetAddress	Physical erase address (aligned to sector size).
	length	Erase length in bytes (aligned to sector size).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The requested job has been accepted by the module. E_NOT_OK: The requested job has not been accepted by the module. E_MEM_SERVICE_NOT_AVAIL: The service function is not implemented.
Description:	Triggers an erase job of the given sector/sector batch defined by targetAddress and length. The result of this service can be retrieved using the Mem_GetJobResult API. If the erase operation was successful, the result of the job is MEM_JOB_OK. If the erase operation failed, e.g. due to a hardware issue, the result of the job is MEM_JOB_FAILED.	
Available via:	Mem_76_Pfls.h	

6.3.2.4 Mem_76_Pfls_BlankCheck

Service name:	Mem_76_Pfls_BlankCheck
Syntax:	Std_ReturnType Mem_76_Pfls_BlankCheck **(** Mem_InstanceIdType instanceId,

	Mem_AddressType targetAddress, Mem_LengthType length**)**	
Service ID[hex]:	0x07	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	instanceId	ID of the related memory driver instance.
	targetAddress	Physical blank check address.
	length	Blank check length.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The requested job has been accepted by the module. E_NOT_OK: The requested job has not been accepted by the module. E_MEM_SERVICE_NOT_AVAIL: The service function is not implemented.
Description:	Triggers a job to check the erased state of the page which is referenced by targetAddress. The result of this service can be retrieved using the Mem_GetJobResult API. If the checked page is blank, the result of the job is MEM_JOB_OK. Otherwise, if the page is not blank, the result is MEM_INCONSISTENT.	
Available via:	Mem_76_Pfls.h	

6.3.2.5 Mem_76_Pfls_HwSpecificService

Service name:	Mem_76_Pfls_HwSpecificService
Syntax:	Std_ReturnType Mem_76_Pfls_HwSpecificService **(** Mem_InstanceIdType instanceId, Mem_HwServiceIdType hwServiceId, Mem_DataType* dataPtr, Mem_LengthType* lengthPtr**)**
Service ID[hex]:	0x0a
Sync/Async:	Asynchronous

Reentrancy:	Non Reentrant	
Parameters (in):	instanceId	ID of the related memory driver instance.
	hwServiceId	Hardware specific service request identifier for dispatching the request.
	dataPtr	Request specific data pointer.
	lengthPtr	Size pointer of the data passed by dataPtr.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The requested job has been accepted by the module. E_NOT_OK: The requested job has not been accepted by the module. E_MEM_SERVICE_NOT_AVAIL: The service function is not implemented.
Description:	Triggers a hardware specific memory driver job. dataPtr can be used to pass and return data to from this service. This service is just a dispatcher to the hardware specific service implementation referenced by hwServiceId. The result of this service can be retrieved using the Mem_GetJobResult API. If the hardware specific operation was successful, the result of the job is MEM_JOB_OK. If the hardware specific operation failed, the result of the job is MEM_JOB_FAILED.	
Available via:	Mem_76_Pfls.h	

6.3.3 Call-back notifications

None

6.3.4 Scheduled Functions

6.3.4.1 Mem_76_Pfls_MainFunction

Service name:	Mem_76_Pfls_MainFunction
Syntax:	void Mem_76_Pfls_MainFunction (void)

Service ID[hex]:	0x03
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Description:	Service to handle the requested jobs and the internal management operations.
Available via:	Mem_76_Pfls.h

7 Generator

7.1 Generator Option

Options	Description
-G,--Generation	Symbolic parameters to be used for fore generation (skip validation).
-H,--Help	Display this help message.
-I,--Input <I>	ECU description file path of the module for which generation tool need to run.
-L,--Log	Symbolic parameters to be used for generation error log.
-M,--Module <M>	Specify module name and version to be generated code for.
-O,--Output <O>	Project-relative path to location where the generated code is to be placed.
-T,--Top_path <T>	Symbolic parameters to be used for set path of module.
-V,--Validate	Symbolic parameters to be used for invoking validation checks.

7.2 Generator Error Message

This section helps to analyze the errors or warnings displayed during the execution of the tool. It ensures conformance of input file(s) with syntax and semantics.

The Generation Tool displays errors or warnings or information when the user has configured incorrect inputs. The format of Error/Warning/Information message is as shown below:

- ERR/WRN/INF<mid><xxx>: < Error/Warning/Information Message>

  Where,
  <mid>: 140 – CanSM Module Id (140) for user configuration checks.
  000 - for command line checks.
  <xxx>: 001 - 999 - Message ID.
- File Name : Name of the file in which the error has occurred
- Path : Absolute path of the container in which the parameter is present

'File Name' and 'Path' are optional.

Below section provides the list of error, warning and information messages.

7.2.1 Error Messages

ERROR040003: The value of parameter \"MemReadBurstSize\" must be aligned with value of parameter \"MemReadPageSize”\.

This error occurs if the configuration value of parameter MemReadBurstSize is different from that of MemReadPageSize.

ERROR040004: The value of parameter \"MemWritePageSize\" in container “short name” must equals container “short name”.

This error occurs if the configuration value of MemWritePageSize of each MemSectorBatch within an instance is different from each others.

ERROR040005: The value of parameter \"MemWriteBurstSize\" in container “short name” must equals container “short name”.

This error occurs if the configuration value of MemWriteBurstSize of each MemWriteBurstSize within an instance is different from each others.

ERROR040006: The value of parameter \"MemStartAddress\" must be aligned with value of parameter \"MemEraseSectorSize\.

This error occurs if the configured value of MemStartAddress is not aligned with the configured value of MemEraseSectorSize.

ERROR040007: The value of parameter \"MemStartAddress\" must be aligned with value of parameter \"MemWritePageSize
This error occurs if the configured value of MemStartAddress is not aligned with the configured value of MemWritePageSize.

ERROR040008: The value of parameter \"MemStartAddress\" must be aligned with value of parameter \"MemReadPageSize
This error occurs if the configured value of MemStartAddress is not aligned with the configured value of

MemReadPageSize.

ERROR040009: The value of parameter \"MemErasePageSize\" in container "short name" must equals container "short name".

This error occurs if the configuration value of MemErasePageSize of each contain MemSectorBatch within an instance is different from each others.

ERROR040011: The value of parameter \"MemWriteBurstSize\" must be aligned with value of parameter \"MemWritePageSize\"

This error occurs if the configuration value of MemWriteBurstSize is not aligned with the value of MemWritePageSize within the same instance

8 Appendix

NA

8.1 Supported Mcu

The Mem_76_Pfls module currently support the following MCU
+ S32K311
+ S32K312
+ S32K314
+ TC39X
+ TC38X
+ TC37X
+ TC36X
+ TC35X
+ TC33X
+ TC29X
+ TC27X
+ CYT4BBX
+ CYT4BFX
+ CYT6BJX
+ S32K148
+ S32K146
+ S32K144
+ S32K142
+ RH850F1KM-S4
+ MPC5606B
+ MPC5607B
+ SPC58EC
+ SPC58NH
+ S32G2

8.2 PFlash size

Each MCU has different Pflash size and Block Size

8.2.1 S32K314

PFlash of S32K314 MCU includes 4MB devided by 6 blocks

Block	Size	Address Range
Code Flash Memory 0	1MB	0x0040 0000 – 0x004F FFFF
Code Flash Momory 1	1MB	0x0050 0000 – 0x005F FFFF
Code Flash Momory 2	1MB	0x0060 0000 – 0x006F FFFF
Code Flash Momory 3	1MB	0x0070 0000 – 0x007F FFFF

8.2.2 S32K312

PFlash of S32K312 MCU includes 2MB devided by 2 blocks

Block	Size	Address Range
Code Flash Memory 0	1MB	0x0040 0000 – 0x004F FFFF
Code Flash Momory 1	1MB	0x0050 0000 – 0x005F FFFF

8.2.3 S32K311

PFlash of S32K311 MCU includes 1MB devided by 2 blocks

Block	Size	Address Range
Code Flash Memory 0	512KB	0x0040 0000 – 0x0047 FFFF
Code Flash Memory 1	512KB	0x0050 0000 – 0x0057 FFFF

8.2.4 TC39X

PFlash of TC39X MCU includes 16MB devided by 6 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	3 Mbyte	0x8000 0000 – 0x802F FFFF
Program Flash 1 (PF1)	3 Mbyte	0x8030 0000 – 0x805F FFFF
Program Flash 2 (PF2)	3 Mbyte	0x8060 0000 – 0x808F FFFF
Program Flash 3 (PF3)	3 Mbyte	0x8090 0000 – 0x80BF FFFF
Program Flash 4 (PF4)	3 Mbyte	0x80C0 0000 – 0x80EF FFFF
Program Flash 5 (PF5)	1 Mbyte	0x80F0 0000 – 0x80FF FFFF

8.2.5 TC38X

PFlash of TC38X MCU includes 10MB devided by 4 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	3 Mbyte	0x8000 0000 – 0x802F FFFF
Program Flash 1 (PF1)	3 Mbyte	0x8030 0000 – 0x805F FFFF
Program Flash 2 (PF2)	3 Mbyte	0x8060 0000 – 0x808F FFFF
Program Flash 3 (PF3)	1 Mbyte	0x8090 0000 – 0x809F FFFF

8.2.6 TC37X

PFlash of TC37X MCU includes 6MB devided by 2 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	3 Mbyte	0x8000 0000 – 0x802F FFFF
Program Flash 1 (PF1)	3 Mbyte	0x8030 0000 – 0x805F FFFF

8.2.7 TC36X

PFlash of TC36X MCU includes 4MB devided by 2 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	2 Mbyte	0x8000 0000 – 0x801F FFFF
Program Flash 1 (PF1)	2 Mbyte	0x8030 0000 – 0x804F FFFF

8.2.8 TC35X

PFlash of TC35X MCU includes 4MB devided by 2 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	2 Mbyte	0x8000 0000 – 0x801F FFFF
Program Flash 1 (PF1)	2 Mbyte	0x8030 0000 – 0x804F FFFF

8.2.9 TC33X

PFlash of TC33X MCU includes 3MB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	3 Mbyte	0x8000 0000 – 0x802F FFFF

8.2.10 TC29X

PFlash of TC29X MCU includes 8MB devided by 4 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	2 Mbyte	0x8000 0000 – 0x801F FFFF
Program Flash 1 (PF1)	2 Mbyte	0x8020 0000 – 0x803F FFFF
Program Flash 2 (PF2)	2 Mbyte	0x8040 0000 – 0x805F FFFF
Program Flash 3 (PF3)	2 Mbyte	0x8060 0000 – 0x807F FFFF

8.2.11 TC27X

PFlash of TC27X MCU includes 4MB devided by 2 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	2 Mbyte	0x8000 0000 – 0x801F FFFF
Program Flash 1 (PF1)	2 Mbyte	0x8020 0000 – 0x803F FFFF

8.2.12 CYT4BBX

PFlash of CYT4BBX MCU includes 4MB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	4 Mbyte	0x1000 0000 - 0x1040 FFFF

8.2.13 CYT4BFX

PFlash of CYT4BFX MCU includes 8MB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	8 Mbyte	0x1000 0000 - 0x1082 FFFF

8.2.14 CYT6BJX

PFlash of CYT6BJX MCU includes 16MB devided by 2 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	8 Mbyte	0x1000 0000 - 0x1082 FFFF
Program Flash 1 (PF1)	8 Mbyte	0x1800 0000 - 0x1882 FFFF

8.2.15 S32K148

PFlash of S32K148 MCU includes 1.5MB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	1.5 Mbyte	0x0000 0000 – 0x0017 FFFF

8.2.16 S32K144

PFlash of S32K144 MCU includes 512KB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	512KB	0x0000 0000 – 0x0007 FFFF

8.2.17 S32K146

PFlash of S32K146 MCU includes 1MB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	1MB	0x0000 0000 – 0x000F FFFF

8.2.18 S32K142

PFlash of S32K142 MCU includes 256KB devided by 1 blocks

Block	Size	Address Range
Program Flash 0 (PF0)	256KB	0x0000 0000 – 0x0003 FFFF

8.2.19 RH850F1KM-S4

PFlash of RH850F1KM-S4 MCU includes 4MB devided by 1 blocks

Block	Size	Address Range
Program Flash	4MB	0x0000 0000 – 0x003D 7FFF

8.2.20 MPC5607B

PFlash of MPC5607B MCU

Block	Size	Address Range
Code flash memory array 0	32 Kbyte	0x0000 0000 – 0x0000 7FFF
Code flash memory array 0	16 Kbyte	0x0000 8000 – 0x0000 BFFF
Code flash memory array 0	16 Kbyte	0x0000 C000 – 0x0000 FFFF
Code flash memory array 0	32 Kbyte	0x0001 0000 – 0x0001 7FFF
Code flash memory array 0	32 Kbyte	0x0001 8000 – 0x0001 FFFF
Code flash memory array 0	128 Kbyte	0x0002 0000 – 0x0003 FFFF
Code flash memory array 0	128 Kbyte	0x0004 0000 – 0x0005 FFFF
Code flash memory array 0	128 Kbyte	0x0006 0000 – 0x0007 FFFF
Code flash memory array 1	128 Kbyte	0x0008 0000 – 0x0009 FFFF
Code flash memory array 1	128 Kbyte	0x000A 0000 – 0x000B FFFF
Code flash memory array 1	128 Kbyte	0x000C 0000 – 0x000D FFFF
Code flash memory array 1	128 Kbyte	0x000E 0000 – 0x000F FFFF
Code flash memory array 2	128 Kbyte	0x0010 0000 – 0x0011 FFFF
Code flash memory array 2	128 Kbyte	0x0012 0000 – 0x0013 FFFF
Code flash memory array 2	128 Kbyte	0x0014 0000 – 0x0015 FFFF

Code flash memory array 2	128 Kbyte	0x0016 0000 – 0x0017 FFFF

8.2.21 MPC5606B

PFlash of MPC5606B MCU

Block	Size	Address Range
Code flash memory array 0	32 Kbyte	0x0000 0000 – 0x0000 7FFF
Code flash memory array 0	16 Kbyte	0x0000 8000 – 0x0000 BFFF
Code flash memory array 0	16 Kbyte	0x0000 C000 – 0x0000 FFFF
Code flash memory array 0	32 Kbyte	0x0001 0000 – 0x0001 7FFF
Code flash memory array 0	32 Kbyte	0x0001 8000 – 0x0001 FFFF
Code flash memory array 0	128 Kbyte	0x0002 0000 – 0x0003 FFFF
Code flash memory array 0	128 Kbyte	0x0004 0000 – 0x0005 FFFF
Code flash memory array 0	128 Kbyte	0x0006 0000 – 0x0007 FFFF
Code flash memory array 1	128 Kbyte	0x0008 0000 – 0x0009 FFFF
Code flash memory array 1	128 Kbyte	0x000A 0000 – 0x000B FFFF
Code flash memory array 1	128 Kbyte	0x000C 0000 – 0x000D FFFF
Code flash memory array 1	128 Kbyte	0x000E 0000 – 0x000F FFFF

8.2.22 SPC58EC

PFlash of SPC58EC MCU

Block	Size	Address Range
B1F0	16 Kbyte	0x00FC 0000 – 0x00FC 7FFF
B1F1	16 Kbyte	0x00FC 8000 – 0x00FC BFFF
B1F2	32 Kbyte	0x00FD 8000 – 0x00FC FFFF
B1F3	64 Kbyte	0x0061 C000 – 0x0061 FFFF
B1F4	64 Kbyte	0x0062 0000 – 0x0062 FFFF
B1F5	128 Kbyte	0x0102 8000 – 0x0103 FFFF
B1F6	256 Kbyte	0x0120 0000 – 0x0123 FFFF
B1F7	256 Kbyte	0x0124 0000 – 0x0127 FFFF
B1F8	256 Kbyte	0x0128 0000 – 0x012B FFFF
B1F9	256 Kbyte	0x012C 0000 – 0x012F FFFF
B1F10	256 Kbyte	0x0130 0000 – 0x0133 FFFF
B1F11	256 Kbyte	0x0134 0000 – 0x0137 FFFF
B1F12	256 Kbyte	0x0138 0000 – 0x013B FFFF

8.2.23 SPC58NH

PFlash of SPC58NH MCU

Block	Size	Address Range
B1F0	16 Kbyte	0x00FC 0000 – 0x00FC 7FFF
B1F1	16 Kbyte	0x00FC 8000 – 0x00FC BFFF
B1F2	32 Kbyte	0x00FD 8000 – 0x00FC FFFF
B1F3	64 Kbyte	0x0061 C000 – 0x0061 FFFF
B1F4	64 Kbyte	0x0062 0000 – 0x0062 FFFF
B1F5	128 Kbyte	0x0102 8000 – 0x0103 FFFF
B1F6	256 Kbyte	0x0120 0000 – 0x0123 FFFF
B1F7	256 Kbyte	0x0124 0000 – 0x0127 FFFF
B1F8	256 Kbyte	0x0128 0000 – 0x012B FFFF

B1F9	256 Kbyte	0x012C 0000 – 0x012F FFFF
B1F10	256 Kbyte	0x0130 0000 – 0x0133 FFFF
B1F11	256 Kbyte	0x0134 0000 – 0x0137 FFFF
B1F12	256 Kbyte	0x0138 0000 – 0x013B FFFF
B1F13	256 Kbyte	0x013C 0000 – 0x013F FFFF
B1F14	256 Kbyte	0x0140 0000 – 0x0143 FFFF
B1F15	256 Kbyte	0x0144 0000 – 0x0147 FFFF
B1F16	256 Kbyte	0x0148 0000 – 0x014B FFFF
B1F17	256 Kbyte	0x014C 0000 – 0x014F FFFF
B1F18	256 Kbyte	0x0150 0000 – 0x0153 FFFF
B1F19	256 Kbyte	0x0154 0000 – 0x0157 FFFF
B1F20	256 Kbyte	0x0158 0000 – 0x015B FFFF
B1F21	256 Kbyte	0x015C 0000 – 0x0159 FFFF
B1F22	256 Kbyte	0x0160 0000 – 0x0163 FFFF
B1F23	256 Kbyte	0x0164 0000 – 0x0167 FFFF
B1F24	256 Kbyte	0x0168 0000 –0x016B FFFF
B1F25	256 Kbyte	0x016C 0000–0x0169 FFFF
B1F26	256 Kbyte	0x0170 0000– 0x0173 FFFF
B1F27	256 Kbyte	0x0174 0000– 0x0177 FFFF
B1F28	256 Kbyte	0x0178 0000– 0x017B FFFF
B1F29	256 Kbyte	0x017C 0000 – 0x0179 FFFF
B1F30	256 Kbyte	0x0180 0000 – 0x0183 FFFF
B1F31	256 Kbyte	0x0184 0000 – 0x0187 FFFF
B1F32	256 Kbyte	0x0188 0000 – 0x018B FFFF
B1F33	256 Kbyte	0x018C 0000 – 0x0189 FFFF
B1F34	256 Kbyte	0x0190 0000 – 0x0193 FFFF
B1F35	256 Kbyte	0x0194 0000 – 0x0197 FFFF
B1F36	256 Kbyte	0x0198 0000 – 0x019C 0000

8.2.24 S32G2

PFlash of S32G2 MCU includes 512MB devided by 1 blocks

Block	**Size**	**Address Range**
Program Flash	512MB	0x0000 0000 - 0x2000 0000