

User Manual

for S32K3 MEM_EEP Driver

Document Number: UM34MEM_EEP ASRR21-11 Rev0000R3.0.0 Rev. 1.0

1 Revision History	2
2 Introduction	3
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
3 Driver	7
3.1 Requirements	7
3.2 Driver Design Summary	7
3.2.1 Overview	7
3.2.2 Features	8
3.2.3 Abort asynchronous transactions:	9
3.3 Hardware Resources	9
3.4 Deviations from Requirements	9
3.5 Driver Limitations	11
3.6 Driver usage and configuration tips	11
3.6.1 Card types selection	11
3.6.2 Operating Voltage Selection	12
3.6.3 DMA modes	14
3.6.4 Compare feature in Mem_43_Eep_HwSpecificService() API.	15
3.7 Runtime errors	17
3.8 Symbolic Names Disclaimer	17
4 Tressos Configuration Plug-in	18
4.1 Module Mem	21
4.2 Container MemGeneral	21
4.3 Parameter MemDevErrorDetect	22
4.4 Container MemInstance	22
4.5 Parameter MemInstanceId	22
4.6 Container MemSectorBatch	23
4.7 Parameter MemEraseSectorSize	23
4.8 Parameter MemNumberOfSectors	24
4.9 Parameter MemReadPageSize	24
4.10 Parameter MemSpecifiedEraseCycles	25
4.11 Parameter MemStartAddress	25
4.12 Parameter MemWritePageSize	26
4.13 Container MemBurstSettings	26
4.14 Parameter MemEraseBurstSize	26

4.15 Parameter MemReadBurstSize	27
4.16 Parameter MemWriteBurstSize	27
4.17 Container MemPublishedInformation	28
4.18 Parameter MemErasedValue	28
4.19 Container Mem_EepUsdhcControllerCfg	29
4.20 Parameter cardDetectDat3	29
4.21 Parameter EnableAutoCMD12	30
4.22 Parameter Mem_EepSynchronizeCache	30
4.23 Parameter Mem_EepUsdhcIpDevErrorDetect	31
4.24 Parameter endianMode	32
4.25 Parameter EnableADMA1Mode	32
4.26 Parameter dmaMode	33
4.27 Parameter admaTable	33
4.28 Parameter admaTableSize	34
4.29 Parameter WriteWatermarkLevel	34
4.30 Parameter ReadWatermarkLevel	35
4.31 Parameter Mem_EepTimeoutMethod	35
4.32 Parameter cardActiveTimeout	36
4.33 Parameter SwitchVoltageTimeout	36
4.34 Parameter GetPresentStatusTimeout	37
4.35 Parameter AbortTransferTimeout	37
4.36 Parameter setBusClkTimeout	38
4.37 Parameter ResetUsdhcTimeout	38
4.38 Parameter transferSyncTimeout	39
4.39 Parameter transferAsyncTimeout	39
4.40 Parameter getInterfaceTimeout	39
4.41 Parameter cardInsertCallback	40
4.42 Parameter cardRemoveCallback	40
4.43 Parameter transferCompleteCallback	41
4.44 Parameter transferErrorCallback	41
4.45 Parameter blockGapCallback	42
4.46 Parameter cardIntCallback	42
4.47 Container voltageSupport	42
4.48 Parameter Support33V	43
4.49 Parameter Support30V	43
4.50 Parameter Support18V	44
4.51 Container BootHostConfig	44
4.52 Parameter ackFlag	44
4.53 Parameter ackTimeout	45
4.54 Parameter disChkTimeout	46

4.55 Parameter autoBlkGap	46
4.56 Parameter ddrEnable	47
4.57 Parameter bootMode	47
4.58 Parameter bootBusWidth	48
4.59 Parameter stopBlockGapCount	48
4.60 Container Mem_EepSdCfg	48
4.61 Parameter Mem_EepEnableOperationsOnEmmc	49
4.62 Parameter Mem_EepBusWidthMode	49
4.63 Parameter Mem_EepCardDriverStrength	50
4.64 Parameter Mem_EepSdIpDevErrorDetect	50
4.65 Parameter afrequencyTable	51
4.66 Parameter Mem_EepSdAutoDiscoverFrequency	51
4.67 Parameter u32McuClock	52
4.68 Parameter SdMaxVoltRetries	52
4.69 Parameter SdMaxCmd55Retries	53
4.70 Parameter SdMaxSendStatusRetries	53
4.71 Reference usdhcCfgReference	54
4.72 Reference SdClockReference	54
4.73 Container BootCardConfig	55
4.74 Parameter bootPartitionSelect	55
4.75 Parameter ackEnable	55
4.76 Parameter bootSpeed	56
4.77 Parameter bootBusWidth	56
4.78 Container AutosarExt	57
4.79 Parameter Mem_EepSuspendApi	57
4.80 Parameter Mem_EepResumeApi	58
4.81 Parameter Mem_EepPropageErrorApi	58
4.82 Parameter Mem_EepEraseApi	59
4.83 Parameter Mem_EepReadApi	59
4.84 Parameter Mem_EepWriteApi	60
4.85 Parameter Mem_EepBankCheckApi	60
4.86 Parameter Mem_EepServiceCompare	61
4.87 Parameter Mem_EepGetJobResultApi	61
4.88 Parameter Mem_EepVersionInfoApi	62
4.89 Parameter Mem_EepBlockSizeWriteAsynchBehaviorEn	62
4.90 Parameter Mem_EepBlockSizeEraseAsynchBehaviorEn	63
4.91 Parameter Mem_EepEnableUserModeSupport	63
4.92 Container Mem_EepInitConfiguration	64
4.93 Parameter Mem_EepMaxReadBlockSize	64
4.94 Parameter Mem_EepMaxWriteBlockSize	65

4.95 Container CommonPublishedInformation	65
4.96 Parameter ArReleaseMajorVersion	65
4.97 Parameter ArReleaseMinorVersion	66
4.98 Parameter ArReleaseRevisionVersion	66
4.99 Parameter ModuleId	67
4.100 Parameter SwMajorVersion	67
4.101 Parameter SwMinorVersion	68
4.102 Parameter SwPatchVersion	68
4.103 Parameter VendorApiInfix	69
4.104 Parameter VendorId	69
5 Module Index	71
5.1 Software Specification	71
6 Module Documentation	72
6.1 Mem_43_Eeprom Driver	72
6.1.1 Detailed Description	72
6.1.2 Data Structure Documentation	74
6.1.3 Macro Definition Documentation	76
6.1.4 Types Reference	78
6.1.5 Enum Reference	79
6.1.6 Function Reference	81
6.2 SD_EMMC	82
6.2.1 Detailed Description	82
6.2.2 Data Structure Documentation	91
6.2.3 Macro Definition Documentation	98
6.2.4 Types Reference	143
6.2.5 Enum Reference	143
6.3 uSDHC	153
6.3.1 Detailed Description	153
6.3.2 Data Structure Documentation	157
6.3.3 Macro Definition Documentation	158
6.3.4 Types Reference	169
6.3.5 Enum Reference	170
6.3.6 Variable Documentation	174



Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	S32K3 Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 3.0.0

Chapter 2

Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This User Manual describes NXP Semiconductor AUTOSAR MEM_43_EEP for S32K3. AUTOSAR MEM_43↔_EEP driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR MEM_43_EEP driver requirements and APIs are described in the AUTOSAR MEM_43↔_EEP driver software specification document.

2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k310_mqfp100
- s32k310_lqfp48
- s32k311_mqfp100 / MWCT2015S_mqfp100
- s32k311_lqfp48
- s32k312_mqfp100 / MWCT2016S_mqfp100
- s32k312_mqfp172 / MWCT2016S_mqfp172
- s32k314_mqfp172
- s32k314_mapbga257
- s32k322_mqfp100 / MWCT2D16S_mqfp100
- s32k322_mqfp172 / MWCT2D16S_mqfp172

- s32k324_mqfp172 / MWCT2D17S_mqfp172
- s32k324_mapbga257
- s32k341_mqfp100
- s32k341_mqfp172
- s32k342_mqfp100
- s32k342_mqfp172
- s32k344_mqfp172
- s32k344_mapbga257
- s32k394_mapbga289
- s32k396_mapbga289
- s32k358_mqfp172
- s32k358_mapbga289
- s32k328_mqfp172
- s32k328_mapbga289
- s32k338_mqfp172
- s32k338_mapbga289
- s32k348_mqfp172
- s32k348_mapbga289
- s32m274_lqfp64
- s32m276_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
DET	Default Error Tracer
DEM	Diagnostic Event Manager
ECC	Error Correcting Code
VLE	Variable Length Encoding
N/A	Not Available
MCU	Microcontroller Unit
ECU	Electronic Control Unit
MEM_43_EEPROM	Electrically Erasable Programmable Read-Only Memory
EA	MEM_43_EEPROM Abstraction
USDHC	Ultra Secured Digital Host Controller
EMMC	Enhanced Multi Media Card
MEM_43_EEP	EEPROM driver
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of MEM Driver	AUTOSAR Release R21-11
2	Reference Manual	S32K3xx Reference Manual, Rev.6, Draft B, 01/2023
		S32K39 and S32K37 Reference Manual, Rev. 2 Draft A, 11/2022
3	Datasheet	S32K3xx Data Sheet, Rev. 6, 11/2022
		S32K396 Data Sheet, Rev. 1.1 — 08/2022
4	Errata	S32K358_0P14E Mask Set Errata — Rev. 28, 9/2022
		S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022
		S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023
		S32K312: Mask Set Errata for Mask 0P09C, Rev. 25/April/2022
		S32K342: Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022
		S32K3x4: Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/Oct/2022

Chapter 3

Driver

- [Requirements](#)
- [Driver Design Summary](#)
- [Hardware Resources](#)
- [Deviations from Requirements](#)
- [Driver Limitations](#)
- [Driver usage and configuration tips](#)
- [Runtime errors](#)
- [Symbolic Names Disclaimer](#)

3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR R21-11 Mem_43_Eep Driver Software Specification document (See Table [Reference List](#))

3.2 Driver Design Summary

This chapter provides the information regarding Mem_43_Eep integration with the uSDHC stack

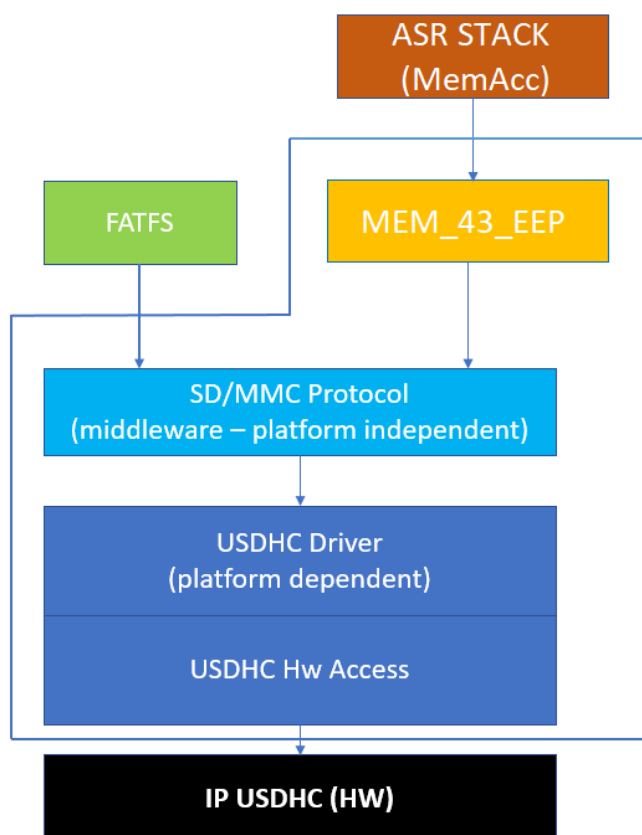
3.2.1 Overview

The Mem_43_Eep driver software consists of three layers:

Driver

#	Layer	Description
1	Mem_43_Eep ASR (high level layer)	Provides Autosar Specification Compliant APIs for reading, writing, erasing, comparing to/from Mem_43_Eep Device
2	SD/eMMC driver (middleware)	Provides services for reading, writing and erasing SD/eMMC device memory
3	uSDHC driver (low level layer)	Implements initialization of low level hardware controller

Software architecture:



3.2.2 Features

3.2.2.1 HLD

- Easy integration in the Autosar MemStack

3.2.2.2 SD/eMMC

- Provide API for accessing SD bus (Host interface, protocol commands interface) via uSDHC driver and for accomplishing uSDHC driver & FatFs integration
- Designed to work with SD memory, miniSD memory, MMC/eMMC cards from any vendors
- Conforms to the SD Host Controller Standard Specification version 3.0
- Supports single block, multi-block read, write and erase

Various configuration options to support for:

- Bus width and data rate selection
- Auto discovery clock frequency
- Retry sending commands for safety purposes

3.2.2.3 uSDHC

- Configuration of the hardware controller
- DMA modes selection
- Supports cache synchronization
- Operating voltage selection
- Callbacks and card detection
- Timeout settings for safety purposes

3.2.3 Abort asynchronous transactions:

- For write: the driver (Mem_43_Eep_Suspend) uses the Stop at block gap request followed by an abort command (CMD12)
- For erase: there is no way to abort the current erase operation, the driver will wait for its completion

3.3 Hardware Resources

The MEM_43_EEP driver uses the USDHC hardware resources.

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR MEM_43_EEP Driver software specification in some places. The table below identifies the AUTOSAR requirements that are not implemented or out of scope for MEM_43_EEP Driver.

Term	Definition
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently or out of scope for the MEM_43_EEP driver.

Requirement	Status	Description	Notes
SWS_Mem_00062	N/I	If the memory hardware provides ECC information, the Mem driver shall check for correctable ECC errors and set the job result code to MEM←_ECC_CORRECTED and proceed with the current job processing.	Can not implement. Not fulfill this requirement.
SWS_Mem_00063	N/I	If the memory hardware provides ECC information, the Mem driver shall check for uncorrectable ECC errors and set the job result code to MEM←_ECC_UNCORRECTED and abort the current job processing.	Can not implement. Not fulfill this requirement.
SWS_Mem_00077	N/I	In case the last memory operation was completed but the ECC circuit corrected an ECC error, the job result shall be set to MEM_ECC_←CORRECTED.	Can not implement. Not fulfill this requirement.
SWS_Mem_00078	N/I	In case the last memory operation didn't complete due to an uncorrectable ECC error, the job result shall be set to MEM_ECC_←UNCORRECTED.	Can not implement. Not fulfill this requirement.
SWS_Mem_00080	N/I	The service Mem_ shall suspend any ongoing flash operations using an according hardware mechanism.	SD hardware only supports suspend/resume for SDIO cards.
SWS_Mem_00081	N/I	The service Mem_Resume shall resume a flash operations that was suspended by the service Mem_Suspend.	SD hardware only supports suspend/resume for SDIO cards.
SWS_Mem_00083	N/I	In case a suspend operation is already in pending, Mem_Suspend shall reject the request by returning E_NOT_OK without further actions.	SD hardware only supports suspend/resume for SDIO cards.
SWS_Mem_00084	N/I	In case no suspend operation is pending, Mem_Resume shall reject the request by returning E_NOT_OK without further actions.	SD hardware only supports suspend/resume for SDIO cards.

Requirement	Status	Description	Notes
SWS_Mem_00085	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_↔Suspend shall check that the Mem driver has been initialized. If this check fails, Mem_Suspend shall raise the development error MEM_E_UNINIT.	SD hardware only supports suspend/resume for SDIO cards.
SWS_Mem_00086	N/I	If development error detection is enabled by MemGeneral.MemDev↔ErrorDetect, the service Mem_↔Resume shall check that the Mem driver has been initialized. If this check fails, Mem_Resume shall raise the development error MEM_E_UNINIT.	SD hardware only supports suspend/resume for SDIO cards.

3.5 Driver Limitations

- Only operations with block size of 512 Bytes are supported.
- For SD card, only SDR25 high speed mode (with 4-bit data buswidth) is supported.
- There is an issue when run with Fast Internal RC Oscillator (FIRC).

3.6 Driver usage and configuration tips

3.6.1 Card types selection

The Mem_43_Eep driver supports two types of cards are SD and eMMC Users have to configure the correct card type by the enabling/disabling the option **Mem_43_Eep Enable Emmc Memory**

- OFF: using SD cards
 - The default and only mode is 4 data bits SDR25 at maximum clock of 52MHz
- ON: using eMMC cards
 - Users can select between various configurations:
 - * Data bus width: 1, 4 or 8 data bits
 - * Data rate: single (SDR) or dual (DDR)
 - * Card driver strength: value of Selected **Driver Strength**[4:7] field in the **HS_TIMING**[185] of Extended CSD register

Mem_43_EepSdCfg

Name

Mem_43_Eep SD Configuration

Mem_43_Eep Enable Emmc Memory* ☐

Mem_43_Eep MMC Buswidth Mode

Card Driver Strength (0 -> 4)

Mem_43_EepSdCfg

Name

Mem_43_Eep SD Configuration

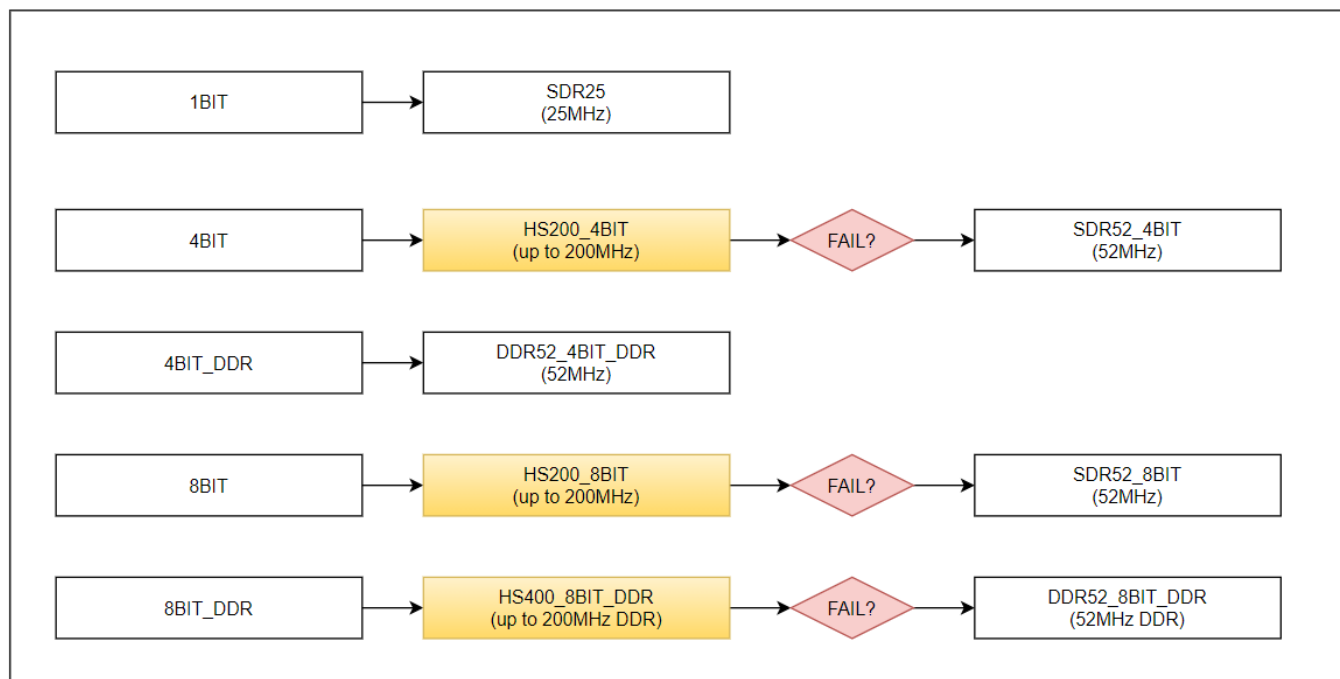
Mem_43_Eep Enable Emmc Memory* ☒

Mem_43_Eep MMC Buswidth Mode

Card Driver Strength (0 -> 4)

SD cards

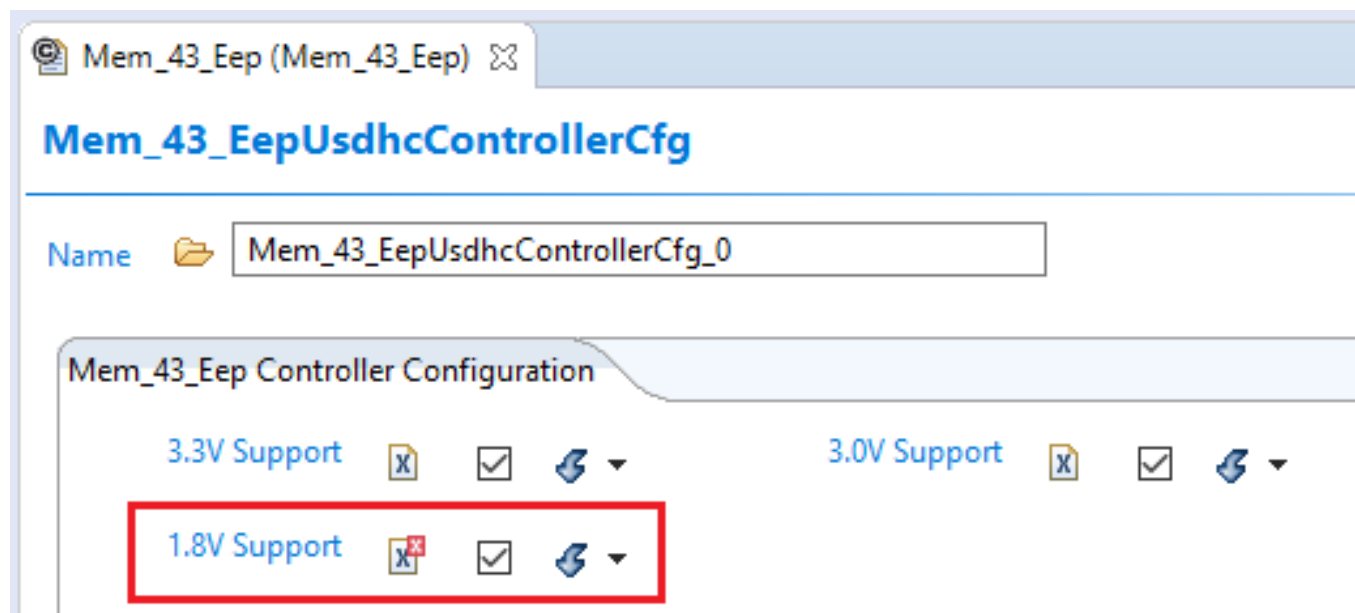
eMMC cards



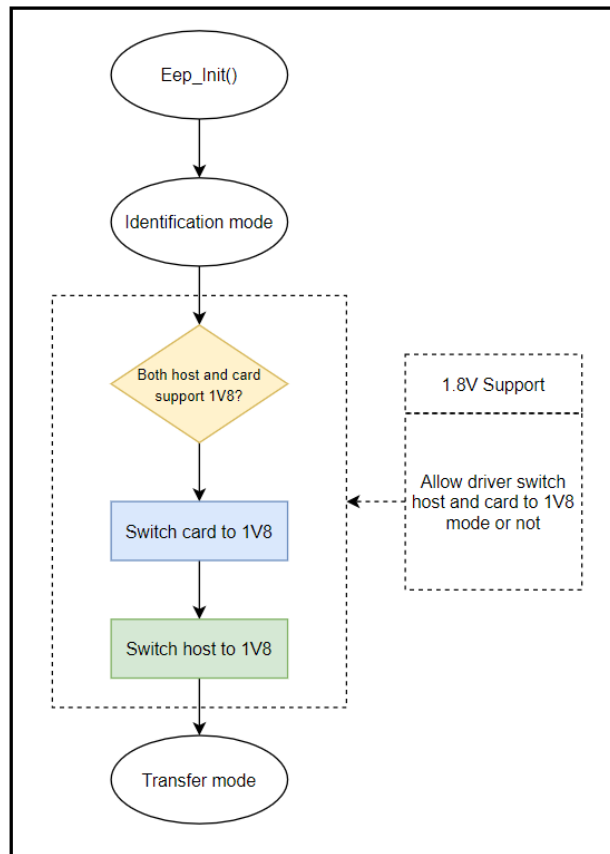
3.6.2 Operating Voltage Selection

- Now adays, most uSDHC controller and SD/eMMC cards support to run at low power mode 1V8.
- During initialization, the driver code will check the 1V8 is supported by both the host and card and switch to that mode if possible
- In some circumstances, the board might not have the 1V8 supply, to deal with this, the Mem_43_Eep driver provides options to adapt with this situation

The GUI configuration:



The 1V8 switching code flow:




3.6.3 DMA modes

The uSDHC host controller implements two types of ADMA: ADMA1 and ADMA2

- ADMA1 data transfer must be 4KB aligned
- ADMA2 eliminates the restriction so that the data transfer just needs to align with 4-byte

Mem_43_EepUsdhcControllerCfg

Name  Mem_43_EepUsdhcControllerCfg_0

Mem_43_Eep Controller Configuration

Use DAT3 pin as a card detection pin

Mem_43_Eep Synchronize Cache

Endian mode

Enable ADMA1 mode




Dma Mode





ADMA table



Size of ADMA table (bytes) (8 -> 65535)



Write Watermark Level (words) (1 -> 128)



Read Watermark Level (words) (1 -> 128)


 ☒  Mem_43_Eep Enable Auto CMD12  ☒


 ☐  Mem_43_Eep Ip Dev Error Detect  ☒ 


 USDHC_ENDIAN_MODE_HALF_WORD_BIG 



 ☐ 



 USDHC_DMA_MODE_NO_DMA 

 USDHC_DMA_MODE_ADMA1

 USDHC_DMA_MODE_ADMA2

 8

 128 

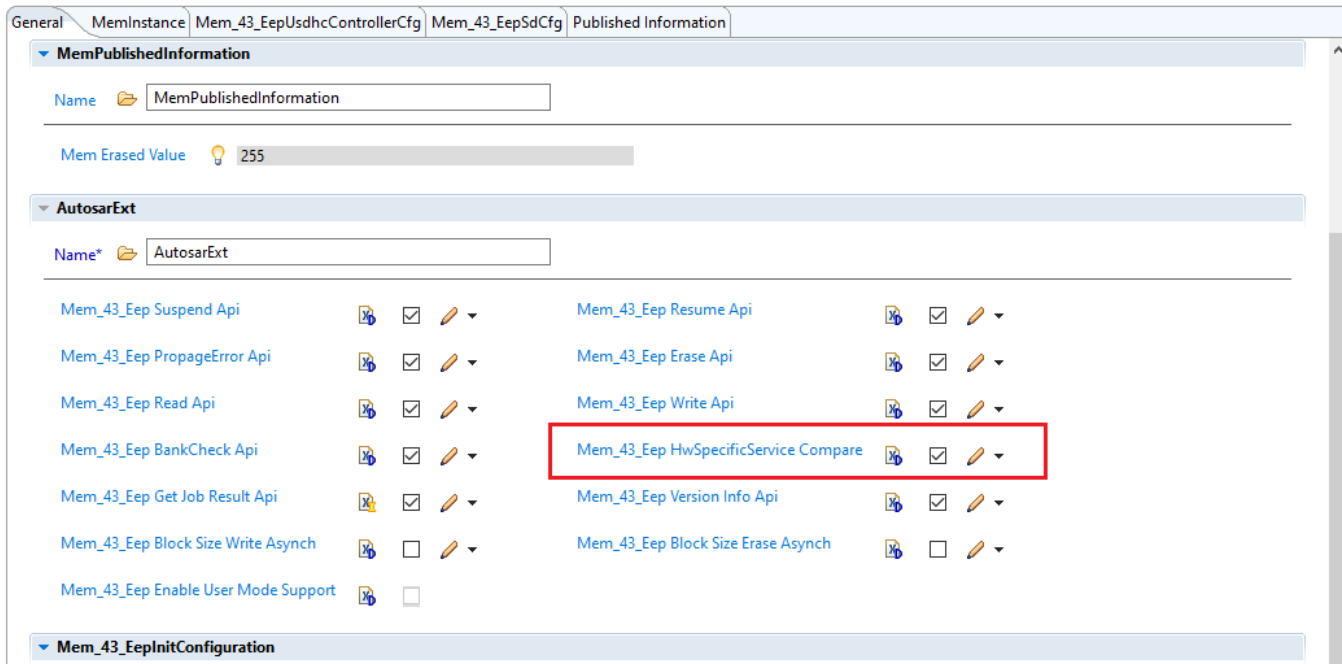
 16 

Note

When ADMA is used, cache coherency problems might appear, the **Synchronize Cache** feature can be used. For more information, please see the chapter **Data Cache Restrictions** in the Integration manual

3.6.4 Compare feature in Mem_43_Eep_HwSpecificService() API.

- Currently HwSpecificService() supports Compare feature with serviceID defined as follows:
 - Define `MEM_43_EEP_HWSERVICEID_COMPARE` 0U.
- To use the Compare feature, users need to enable node "Mem_43_Eep HwSpecificService Compare" on the interface.



- The compare feature needs to set 4 input parameters to the `HwSpecificService()` function and 1 global variable that needs to be set before calling this API.
 - 4 input parameters:
 - * `Mem_HwSpecificService(instanceId, hwServiceId, dataPtr, lengthPtr);`
 - * `Mem_43_Eep_InstanceIdType instanceId`: ID of the related memory driver
 - * `Mem_43_Eep_HwServiceIdType hwServiceId`: `MEM_43_EEP_HWSERVICEID_←_COMPARE` or `0x0U`
 - * `Mem_43_Eep_DataType* dataPtr`: Data pointer to compare
 - * `Mem_43_Eep_LengthType* lengthPtr`: The size pointer of the data to be compared is passed by `dataPtr`.
 - The global variable needs to be set to its value before calling `HwSpecificService()` with the compare feature:
 - * `Mem_43_Eep_AddressType Mem_43_Eep_Conpare_u32EepromAddrNeedsCompared←`: Eep memory address needs to be compared.
- `Mem_43_Eep_Conpare_u32EepromAddrNeedsCompared` needs to be a multiple of 512.
- Default value of `Mem_43_Eep_Conpare_u32EepromAddrNeedsCompared` = `0U` if not set.
- When calling `Mem_43_Eep_HwSpecificService()` with service ID is `MEM_43_EEP_HWSERVICEID_←_COMPARE`, this API just sets the job. It will return `MEM_43_EEP_E_NOT_OK` if the job set is successful and `MEM_43_EEP_E_NOT_OK` if the job cannot be set. The actual Compare job will be done in the `Mem_43_Eep_MainFunction()` function.
- The comparison result is returned when calling `Mem_43_Eep_GetJobResult()`.
 - If both data blocks are the same: return: `MEM_43_EEP_JOB_OK`.
 - If both data blocks are not identical, return: `MEM_43_EEP_INCONSISTENT`.
 - If the execution of reading data from Eep memory fails, return: `MEM_43_EEP_JOB_FAILED`.
- For example:

```

/* Set Eep memory address needs to be compared */
Mem_43_Eep_Conpare_u32EepromAddrNeedsCompared = 0x1000;
/* Set job for Compare feature */
Status = Mem_43_Eep_HwSpecificService(Instance0, MEM_43_EEP_HWSERVICEID_COMPARE, pData, pLength);
if(E_OK == Status)
{
    /* Perform the Compare job */
    while (MEM_43_EEP_JOB_PENDING == Mem_43_Eep_GetJobResult(Instance0))
    {
        Mem_43_Eep_MainFunction();
    }
}
/* Get the Compare job result */
JobResult = Mem_43_Eep_GetJobResult(Instance0);

```

3.7 Runtime errors

The driver generates the following DET errors at runtime.

Function	Error Code	Condition triggering the error
MEM_43_EEP_E_UNINIT	0x01	API service called with init failed
MEM_43_EEP_E_PARAM_POINTER	0x02	NULL_PTR passed
MEM_43_EEP_E_PARAM_ADDRESS	0x03	TargetAddress is not in range and aligned to first byte of mem_43_eeprom sector
MEM_43_EEP_E_PARAM_LENGTH	0x04	TargetAddress is not in range and aligned to last byte of mem_43_eeprom sector
MEM_43_EEP_E_PARAM_INSTANCE↵_ID	0x05	Invalid instace id
MEM_43_EEP_E_JOB_PENDING	0x06	API service called while driver still busy
MEM_43_EEP_E_TIMEOUT	0x07	The hardware operation did not finish before timeout expired.

3.8 Symbolic Names Disclaimer

All containers having symbolicNameValue set to TRUE in the AUTOSAR schema will generate defines like:

```
#define <Mip>Conf_<Container_ShortName>_<Container_ID>
```

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing #ifdefs arguments).

Chapter 4

Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module [Mem](#)
 - Container [MemGeneral](#)
 - * Parameter [MemDevErrorDetect](#)
 - Container [MemInstance](#)
 - * Parameter [MemInstanceId](#)
 - * Container [MemSectorBatch](#)
 - Parameter [MemEraseSectorSize](#)
 - Parameter [MemNumberOfSectors](#)
 - Parameter [MemReadPageSize](#)
 - Parameter [MemSpecifiedEraseCycles](#)
 - Parameter [MemStartAddress](#)
 - Parameter [MemWritePageSize](#)
 - Container [MemBurstSettings](#)
 - Parameter [MemEraseBurstSize](#)
 - Parameter [MemReadBurstSize](#)
 - Parameter [MemWriteBurstSize](#)
 - Container [MemPublishedInformation](#)
 - * Parameter [MemErasedValue](#)
 - Container [Mem_EepUsdhcControllerCfg](#)
 - * Parameter [cardDetectDat3](#)
 - * Parameter [EnableAutoCMD12](#)
 - * Parameter [Mem_EepSynchronizeCache](#)
 - * Parameter [Mem_EepUsdhcIpDevErrorDetect](#)
 - * Parameter [endianMode](#)
 - * Parameter [EnableADMA1Mode](#)
 - * Parameter [dmaMode](#)
 - * Parameter [admaTable](#)
 - * Parameter [admaTableSize](#)

- * Parameter [WriteWatermarkLevel](#)
- * Parameter [ReadWatermarkLevel](#)
- * Parameter [Mem_EepTimeoutMethod](#)
- * Parameter [cardActiveTimeout](#)
- * Parameter [SwitchVoltageTimeout](#)
- * Parameter [GetPresentStatusTimeout](#)
- * Parameter [AbortTransferTimeout](#)
- * Parameter [setBusClkTimeout](#)
- * Parameter [ResetUsdhcTimeout](#)
- * Parameter [transferSyncTimeout](#)
- * Parameter [transferAsyncTimeout](#)
- * Parameter [getInterfaceTimeout](#)
- * Parameter [cardInsertCallback](#)
- * Parameter [cardRemoveCallback](#)
- * Parameter [transferCompleteCallback](#)
- * Parameter [transferErrorCallback](#)
- * Parameter [blockGapCallback](#)
- * Parameter [cardIntCallback](#)
- * Container [voltageSupport](#)
 - Parameter [Support33V](#)
 - Parameter [Support30V](#)
 - Parameter [Support18V](#)
- * Container [BootHostConfig](#)
 - Parameter [ackFlag](#)
 - Parameter [ackTimeout](#)
 - Parameter [disChkTimeout](#)
 - Parameter [autoBlkGap](#)
 - Parameter [ddrEnable](#)
 - Parameter [bootMode](#)
 - Parameter [bootBusWidth](#)
 - Parameter [stopBlockGapCount](#)
- Container [Mem_EepSdCfg](#)
 - * Parameter [Mem_EepEnableOperationsOnEmmc](#)
 - * Parameter [Mem_EepBusWidthMode](#)
 - * Parameter [Mem_EepCardDriverStrength](#)
 - * Parameter [Mem_EepSdIpDevErrorDetect](#)
 - * Parameter [afrequencyTable](#)
 - * Parameter [Mem_EepSdAutoDiscoverFrequency](#)
 - * Parameter [u32McuClock](#)
 - * Parameter [SdMaxVoltRetries](#)
 - * Parameter [SdMaxCmd55Retries](#)
 - * Parameter [SdMaxSendStatusRetries](#)

- * Reference [usdhcCfgReference](#)
- * Reference [SdClockReference](#)
- * Container [BootCardConfig](#)
 - Parameter [bootPartitionSelect](#)
 - Parameter [ackEnable](#)
 - Parameter [bootSpeed](#)
 - Parameter [bootBusWidth](#)
- Container [AutosarExt](#)
 - * Parameter [Mem_EepSuspendApi](#)
 - * Parameter [Mem_EepResumeApi](#)
 - * Parameter [Mem_EepPropageErrorApi](#)
 - * Parameter [Mem_EepEraseApi](#)
 - * Parameter [Mem_EepReadApi](#)
 - * Parameter [Mem_EepWriteApi](#)
 - * Parameter [Mem_EepBankCheckApi](#)
 - * Parameter [Mem_EepServiceCompare](#)
 - * Parameter [Mem_EepGetJobResultApi](#)
 - * Parameter [Mem_EepVersionInfoApi](#)
 - * Parameter [Mem_EepBlockSizeWriteAsynchBehaviorEn](#)
 - * Parameter [Mem_EepBlockSizeEraseAsynchBehaviorEn](#)
 - * Parameter [Mem_EepEnableUserModeSupport](#)
- Container [Mem_EepInitConfiguration](#)
 - * Parameter [Mem_EepMaxReadBlockSize](#)
 - * Parameter [Mem_EepMaxWriteBlockSize](#)
- Container [CommonPublishedInformation](#)
 - * Parameter [ArReleaseMajorVersion](#)
 - * Parameter [ArReleaseMinorVersion](#)
 - * Parameter [ArReleaseRevisionVersion](#)
 - * Parameter [ModuleId](#)
 - * Parameter [SwMajorVersion](#)
 - * Parameter [SwMinorVersion](#)
 - * Parameter [SwPatchVersion](#)
 - * Parameter [VendorApiInfix](#)
 - * Parameter [VendorId](#)

4.1 Module Mem

Configuration of the Mem_Eep (internal or external mem_eeprom driver) module.

Included containers:

- [MemGeneral](#)
- [MemInstance](#)
- [MemPublishedInformation](#)
- [Mem_EepUsdhcControllerCfg](#)
- [Mem_EepSdCfg](#)
- [AutosarExt](#)
- [Mem_EepInitConfiguration](#)
- [CommonPublishedInformation](#)

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantSupport	false
supportedConfigVariants	VARIANT-PRE-COMPILE

4.2 Container MemGeneral

Container for general configuration parameters of the mem_eeprom driver. These parameters are always pre-compile.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Parameter MemDevErrorDetect

Pre-processor switch to enable and disable development error detection.

true : Development error detection enabled.

false: Development error detection disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	true

4.4 Container MemInstance

This container includes the Mem driver instance specific configuration parameters.

Included subcontainers:

- [MemSectorBatch](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	65535
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.5 Parameter 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.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65535
min	0

4.6 Container MemSectorBatch

Configuration description of a programmable sector or sector batch. Sector batch means that homogenous and coherent sectors can be configured as one MemSector element.

Included subcontainers:

- [MemBurstSettings](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.7 Parameter MemEraseSectorSize

Size of this sector in bytes.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	512
max	4294967295
min	1

4.8 Parameter MemNumberOfSectors

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	1
max	65535
min	1

4.9 Parameter MemReadPageSize

Size of a read page of this sector in bytes.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	512
max	4294967295
min	1

4.10 Parameter MemSpecifiedEraseCycles

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	N/A
max	4294967295
min	0

4.11 Parameter MemStartAddress

Physical start address of the sector (batch).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	N/A
max	9223372036854775807
min	0

4.12 Parameter MemWritePageSize

Size of a write page of this sector in bytes.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	512
max	4294967295
min	1

4.13 Container MemBurstSettings

Container for burst setting configuration parameters of the Mem driver.

A sector burst can be used for improved performance.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.14 Parameter 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.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	4096
max	4294967295
min	1

4.15 Parameter MemReadBurstSize

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	1
max	4294967295
min	1

4.16 Parameter 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.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	1

4.17 Container MemPublishedInformation

Additional published parameters not covered by CommonPublishedInformation container.

Note that these parameters do not have any configuration class setting, since they are published information.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.18 Parameter MemErasedValue

The contents of an erased mem_eeeprom memory cell.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	255
max	255
min	0

4.19 Container Mem_EepUsdhcControllerCfg

Configuration for low layer ip

Included subcontainers:

- [voltageSupport](#)
- [BootHostConfig](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.20 Parameter cardDetectDat3

Set this field to true if user want to use DAT3 pin as a card detection pin

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.21 Parameter EnableAutoCMD12

Enable auto CMD12. If this field is false, user have to send a "CMD12" command to card after each time send/receive data to stop the data transfer. If this field is true, uSDHC will send "CMD12" command automatically after data transfer is end.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.22 Parameter Mem_EepSynchronizeCache

Vendor specific:

Synchronize the memory by cleaning the cache before tranfering.

The MEM_EEP driver needs to maintain the memory coherency by means of three methods:

1. Disable data cache, or

2. Configure the mem_eep region upon which the driver operates, as non-cacheable, or

3. Enable the Mem_EepSynchronizeCache feature.

Depending on the application configuration, one option may be more beneficial than other.

Enabled: The MEM_EEP driver will call Mcl cache API functions in order to clean the cache

before transferring , in order

to ensure that the cache and the modified memory are in sync.

If enabled, the driver will attempt to clean only the modified lines from the cache.

If the size of the region to be clean is greater than half of the cache size, then

the entire cache is clean.

Note: If enabled, the MclEnableCache parameter has to be enabled and the MCL plugin included as a dependency.

Disabled: The upper layers have to ensure that the buffers in write/read/compare functions upon which the driver operates is not cached.

This can be obtained by either disabling the data cache or by configuring the memory region as non-cacheable.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.23 Parameter Mem_EepUsdhcIpDevErrorDetect

Pre-processor switch to enable and disable development error detection for IP layer.

true: Development error detection enabled for Ip layer.

false: Development error detection disabled for Ip layer.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.24 Parameter endianMode

Endian mode

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	USDHC_ENDIAN_MODE_LITTLE
literals	['USDHC_ENDIAN_MODE_BIG', 'USDHC_ENDIAN_MODE_HALF_↔WORD_BIG', 'USDHC_ENDIAN_MODE_LITTLE']

4.25 Parameter EnableADMA1Mode

Enable ADMA1 mode. This mode support data transfer of 4KB aligned data in system memory.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.26 Parameter dmaMode

Dma Mode

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	USDHC_DMA_MODE_NO_DMA
literals	['USDHC_DMA_MODE_NO_DMA', 'USDHC_DMA_MODE_ADMA1', 'USDHC_DMA_MODE_ADMA2']

4.27 Parameter admaTable

ADMA table for transferring with ADMA1 and ADMA2 mode.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	usdhc1_admaTable0

4.28 Parameter admaTableSize

The size of ADMA table in bytes.

For ADMA1, driver needs two tables(8 bytes) per a transfer: one for configure the start address(be 4KB align) and the next table for configure length data;

Following the setup, driver can transfer maximum 61440 bytes(0xF000 -> align with 4096) of data at the same time.

For ADMA2, each ADMA table preempt 8 bytes and able to transfer maximum 65532 bytes(0xFFFFC -> align with 4) of data at the same time; for configuration, the start address is 32-bit boundary.

If you need to transfer more data at the same time, you have to use more ADMA table size. Size of ADMA table in range: [8...65535]

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8
max	65535
min	8

4.29 Parameter WriteWatermarkLevel

The size of Write Watermark Level in word.

For internal DMA it's recommended to use integer multiple of 16.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	128
max	128
min	1

4.30 Parameter ReadWatermarkLevel

The size of Read Watermark Level in word.

For internal DMA it's recommended to use 16.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	16
max	128
min	1

4.31 Parameter Mem_EepTimeoutMethod

Vendor specific: Counter type used in timeout detection for MEM_EEP service request.

Based on selected counter type the timeout value will be interpreted as follows:

OSIF_COUNTER_DUMMY - Counts the number of iterations of the waiting loop. The actual timeout depends on many factors: operation type, compiler optimizations, interrupts or other tasks in the system, etc.

OSIF_COUNTER_SYSTEM - Microseconds.

OSIF_COUNTER_CUSTOM - Defined by user implementation of timing services

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_SYSTEM', 'OSIF_COUNTER_CUSTOM', 'OSIF_COUNTER_DUMMY']

4.32 Parameter cardActiveTimeout

Configures the timeout for card active

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.33 Parameter SwitchVoltageTimeout

Configures the timeout for switch votage

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.34 Parameter GetPresentStatusTimeout

Configures the timeout for switch votage

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.35 Parameter AbortTransferTimeout

Configures the timeout for aborting trasfer

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.36 Parameter setBusClkTimeout

Configures the timeout for bus clock set

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.37 Parameter ResetUsdhcTimeout

Reset the uSDHC module

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.38 Parameter transferSyncTimeout

Configures the timeout for sync transfer

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.39 Parameter transferAsyncTimeout

Configures the timeout for async transfer

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.40 Parameter getInterfaceTimeout

Timeout value to get card interface condition

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	268435456
max	4294967295
min	1

4.41 Parameter cardInsertCallback

Card insertion callback.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.42 Parameter cardRemoveCallback

Card removal callback.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.43 Parameter transferCompleteCallback

Transfer complete callback.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.44 Parameter transferErrorCallback

Transfer error callback.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.45 Parameter blockGapCallback

Block gap callback.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.46 Parameter cardIntCallback

Card interrupt callback.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.47 Container voltageSupport

Voltage levels supported on current board

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.48 Parameter Support33V

Current board supports 3.3V voltage

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.49 Parameter Support30V

Current board supports 3.0V voltage

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.50 Parameter Support18V

Current board supports 1.8V voltage (HS200 and HS400 mode will be disabled if 1.8V is not available)

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.51 Container BootHostConfig

Configure flags in mmc boot

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.52 Parameter ackFlag

Enable the boot ACK

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.54 Parameter disChkTimeout

Disable check timeout

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.55 Parameter autoBlkGap

Enable auto stop at the block gap

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.56 Parameter ddrEnable

Enable Double data rate mode

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.57 Parameter bootMode

Boot Mode

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	USDHC_NORMAL_MMC_BOOT
literals	['USDHC_NORMAL_MMC_BOOT', 'USDHC_ALTER_MMC_BOOT']

4.58 Parameter bootBusWidth

Boot Bus Width

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	USDHC_DATA_BUS_WIDTH_1BIT
literals	['USDHC_DATA_BUS_WIDTH_1BIT', 'USDHC_DATA_BUS_WIDTH_4BIT', 'USDHC_DATA_BUS_WIDTH_8BIT']

4.59 Parameter stopBlockGapCount

Configures the block count for blockgap stop in boot process. The value range of this block count is [0 ... 65535]

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65535
min	0

4.60 Container Mem_EepSdCfg

This container is present for SD card configuration structure.

Included subcontainers:

- [BootCardConfig](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.61 Parameter Mem_EepEnableOperationsOnEmmc

Enable: Operations(write/read/erase/compare) are on the Emmc memory

Disable: Operations(write/read/erase/compare) are on the SD memory

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.62 Parameter Mem_EepBusWidthMode

Select bus width mode for eMMC.

Available modes are 1 bit, 4 bits, 8 bits.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SD_IP_DATABUSWIDTH_4BIT
literals	['SD_IP_DATABUSWIDTH_1BIT', 'SD_IP_DATABUSWIDTH_4BIT', 'SD_IP_DATABUSWIDTH_8BIT', 'SD_IP_DATABUSWIDTH_4BIT_↔DDR', 'SD_IP_DATABUSWIDTH_8BIT_DDR']

4.63 Parameter Mem_EepCardDriverStrength

Vendor specific:

Selected the Driver Strength for card pads in HS_TIMING[185].

(only for eMMC in HS200 or HS400 mode)

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4
min	0

4.64 Parameter Mem_EepSdIpDevErrorDetect

Pre-processor switch to enable and disable development error detection for IP layer.

true: Development error detection enabled for Ip layer.

false: Development error detection disabled for Ip layer.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.65 Parameter afrequencyTable

Vendor specific:

Sd Clock Frequency

Indicates Clock Frequency (in Hz) will be used to transfer data. It must be less than uSDHC source clock. Driver will calculate the maximum frequency which can be used based on this field.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.66 Parameter Mem_EepSdAutoDiscoverFrequency

Automatically calculate the maximum of the bus clock based on the configured frequency.

TRUE : Enable the automatic calculation.

FALSE: Disable the automatic calculation. Bus clock equals to the configured frequency.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.67 Parameter u32McuClock

Vendor specific:

SDHC Source Clock

Indicates source clock for uSDHC in Hz. Derived from MCU configuration.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	0
max	4294967295
min	0

4.68 Parameter SdMaxVoltRetries

Maximum loop count to check the card operation voltage range

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	100
max	4294967295
min	1

4.69 Parameter SdMaxCmd55Retries

Maximum loop count to check the SDHC card for cmd55

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	30
max	255
min	1

4.70 Parameter SdMaxSendStatusRetries

Maximum loop count for Send Status command (CMD13) in Sd_Ip_WaitWriteComplete function

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	65535
max	4294967295
min	1

4.71 Reference usdhcCfgReference

Reference to Usdhc Configuration

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/Mem_43_Eep_TS_T40D34M30I0R0/Mem/Mem_EepUsdhcControllerCfg

4.72 Reference SdClockReference

Reference to a container of the type McuClockReferencePoint, to select an input clock.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↔ Config/McuClockReferencePoint

4.73 Container BootCardConfig

This container is present for eMMC Boot Configuration structure.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

4.74 Parameter bootPartitionSelect

Select boot partition for boot operation.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SD_IP_MMCBOOT_PARTITION_1
literals	['SD_IP_MMCBOOT_PARTITION_1', 'SD_IP_MMCBOOT_PARTITION_2', 'SD_IP_MMCBOOT_USER_AREA']

4.75 Parameter ackEnable

Enable ACK for boot operation.

true: ACK enable.

false: ACK disabled

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.76 Parameter bootSpeed

Select boot speed for boot operation.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	SD_IP_MMCBOOT_SDR_LOW_SPEED
literals	['SD_IP_MMCBOOT_SDR_LOW_SPEED', 'SD_IP_MMCBOOT_SDR_↔ HIGH_SPEED', 'SD_IP_MMCBOOT_DDR']

4.77 Parameter bootBusWidth

Boot Data Bus Width

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SD_IP_MMCBOOT_DBW_1BIT
literals	['SD_IP_MMCBOOT_DBW_1BIT', 'SD_IP_MMCBOOT_DBW_4BIT', 'SD_IP_MMCBOOT_DBW_8BIT']

4.78 Container AutosarExt

Vendor specific: This container contains the global Non-Autosar configuration parameters of the Mem_Eep driver.

This container is a MultipleConfigurationContainer, i.e. this container and

its sub-containers exist once per configuration set.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.79 Parameter Mem_EepSuspendApi

Compile switch to enable and disable the Mem_Eep_Suspend function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.80 Parameter Mem_EepResumeApi

Compile switch to enable and disable the Mem_Eep_Resume function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.81 Parameter Mem_EepPropageErrorApi

Compile switch to enable and disable the Mem_Eep_PropageError function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.82 Parameter Mem_EepEraseApi

Compile switch to enable and disable the Mem_Eep_Erase function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.83 Parameter Mem_EepReadApi

Compile switch to enable and disable the Mem_Eep_Read function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.84 Parameter Mem_EepWriteApi

Compile switch to enable and disable the Mem_Eep_Write function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.85 Parameter Mem_EepBankCheckApi

Compile switch to enable and disable the Mem_Eep_BankCheck function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.86 Parameter Mem_EepServiceCompare

Compile switch to enable and disable the Mem_EepServiceCompare feature.

true: feature supported / feature provided.

false: feature not supported / feature not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.87 Parameter Mem_EepGetJobResultApi

Compile switch to enable and disable the Mem_Eep_GetJobResult function.

true: API supported / function provided.

false: API not supported / function not provided

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.88 Parameter Mem_EepVersionInfoApi

Pre-processor switch to enable / disable the API to read out the modules version information.

true: Version info API enabled.

false: Version info API disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.89 Parameter Mem_EepBlockSizeWriteAsynchBehaviorEn

Vendor specific: Enable asynchronous execution of the write job in the Mem_Eep_MainFunction function which doesn't wait (block) for completion of the block size write operation(s).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.90 Parameter Mem_EepBlockSizeEraseAsynchBehaviorEn

Vendor specific: Enable asynchronous execution of the erase job in the Mem_Eep_MainFunction function which doesn't wait (block) for completion of the block size erase operation(s).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.91 Parameter Mem_EepEnableUserModeSupport

When this parameter is enabled, the MEM_EEP module will adapt to run from User Mode, with the following measures:

configuring REG_PROT for Mem_Eep IPs so that the registers under protection can be accessed from user mode by setting UAA bit in REG_PROT_GCR to 1

for more information and availability on this platform, please see chapter User Mode Support in IM

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.92 Container Mem_EepInitConfiguration

Container for runtime configuration parameters of the mem_eeprom driver.

Implementation Type: Mem_Eep_ConfigType.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.93 Parameter Mem_EepMaxReadBlockSize

The maximum number of bytes to read in one cycle of the mem_eeprom driver's job processing function.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4096
max	4294967295
min	0

4.94 Parameter Mem_EepMaxWriteBlockSize

The maximum number of bytes to write in one cycle of the mem_eeprom driver's job processing function.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4096
max	4294967295
min	0

4.95 Container CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.96 Parameter ArReleaseMajorVersion

Vendor specific: Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.97 Parameter ArReleaseMinorVersion

Vendor specific: Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.98 Parameter ArReleaseRevisionVersion

Vendor specific: Patch version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariant Value	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.99 Parameter ModuleId

Vendor specific: Module ID of this module.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariant Value	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	90
max	90
min	90

4.100 Parameter SwMajorVersion

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

4.101 Parameter SwMinorVersion

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.102 Parameter SwPatchVersion

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION

Property	Value
defaultValue	0
max	0
min	0

4.103 Parameter VendorApiInfix

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

`<ModuleName>_>VendorId>_<VendorApiInfix>`.

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write.

This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43

4.104 Parameter VendorId

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43



Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Mem_43_Eeprom Driver	72
SD_EMMC	82
uSDHC	153

Chapter 6

Module Documentation

6.1 Mem_43_Eeprom Driver

6.1.1 Detailed Description

Data Structures

- struct [Mem_43_Eep_JobRuntimeInfoType](#)
Mem job runtime information Type. [More...](#)
- struct [Mem_43_Eep_SectorBatchType](#)
Sector Batch Type. [More...](#)
- struct [Mem_43_Eep_MemInstanceType](#)
Mem Instance Type. [More...](#)
- struct [Mem_43_Eep_ConfigType](#)
Mem_43_Eep Config Type. [More...](#)

Macros

- `#define MEM_43_EEP_E_UNINIT`
Development error codes (passed to DET).
- `#define MEM_43_EEP_E_PARAM_POINTER`
service ID of error: NULL_PTR passed. (passed to DET)
- `#define MEM_43_EEP_E_PARAM_ADDRESS`
service ID of error: TargetAddress is not in range and aligned to first byte of mem_43_eeprom sector. (passed to DET)
- `#define MEM_43_EEP_E_PARAM_LENGTH`
service ID of error: TargetAddress is not in range and aligned to last byte of mem_43_eeprom sector. (passed to DET)
- `#define MEM_43_EEP_E_PARAM_INSTANCE_ID`
service ID of error: invalid instace id passed. (passed to DET)
- `#define MEM_43_EEP_E_JOB_PENDING`
service ID of error: invalid instace id passed. (passed to DET)

- `#define MEM_43_EEP_E_TIMEOUT`
service ID of error: The hardware operation did not finish before timeout expired. (passed to DET)
- `#define MEM_43_EEP_INIT_ID`
All service IDs (passed to DET).
- `#define MEM_43_EEP_START_SEC_CODE`
Start of Mem_43_Eep section CODE.
- `#define MEM_43_EEP_STOP_SEC_CODE`
Stop of Mem_43_Eep section CODE.

Types Reference

- `typedef uint16 Mem_43_Eep_CrcType`
Mem_43_Eep CRC Type.
- `typedef uint32 Mem_43_Eep_AddressType`
Mem_43_Eep Address Type.
- `typedef uint8 Mem_43_Eep_DataType`
Mem_43_Eep Data Type.
- `typedef uint32 Mem_43_Eep_InstanceIdType`
Mem_43_Eep InstanceId Type.
- `typedef uint32 Mem_43_Eep_LengthType`
Mem_43_Eep Length Type.
- `typedef uint32 Mem_43_Eep_HwServiceIdType`
Mem_43_Eep HwServiceId Type.

Enum Reference

- `enum Mem_43_Eep_JobType`
Type of job currently executed by Mem_43_Eep_MainFunction.
- `enum Mem_43_Eep_BlankCheckType`
Asynchronous job result type.
- `enum Mem_43_Eep_CompareCheckType`
- `enum Mem_43_Eep_CrcDataSizeType`
Size of data to be processeed by CRC.
- `enum Mem_43_Eep_JobResultType`
Asynchronous job result type.

Function Reference

- `void Mem_43_Eep_Init (const Mem_43_Eep_ConfigType *ConfigPtr)`
The function initializes Mem_43_Eep module.
- `void Mem_43_Eep_DeInit (void)`
The function deinitializes Mem_43_Eep module.
- `void Mem_43_Eep_MainFunction (void)`
Service to handle the requested jobs and the internal management operations.

6.1.2 Data Structure Documentation

6.1.2.1 struct Mem_43_Eep_JobRuntimeInfoType

Mem job runtime information Type.

This structure contains runtime information the current processing job of each Mem instance.

Definition at line 173 of file [Mem_43_Eep_InternalTypes.h](#).

Data Fields

Type	Name	Description
Mem_43_Eep_LengthType	Mem_43_Eep_u32MaxRead	Maximum number of bytes to read in one cycle of Mem_43_Eep_MainFunction.
Mem_43_Eep_LengthType	Mem_43_Eep_u32MaxWrite	Maximum number of bytes to write in one cycle of Mem_43_Eep_MainFunction.
Mem_43_Eep_AddressType	Mem_43_Eep_u32EepromAddrIt	Logical address of data block currently processed by Mem_43_Eep_MainFunction.
Mem_43_Eep_LengthType	Mem_43_Eep_u32RemainingLength	Remainin length to be transfered until the end of the job.
Mem_43_Eep_JobResultType	Mem_43_Eep_eJobResult	Result of last mem_43_eeprom module job.
Mem_43_Eep_JobType	Mem_43_Eep_eJob	Type of currently executed job (erase, write, read, or blank)
Mem_43_Eep_HwServiceIdType	Mem_43_Eep_u32HwServiceIdJob	Type of currently executed HwServiceId job (compare) when executing ProcessHwSpecificServiceJob.
Mem_43_Eep_DataType *	Mem_43_Eep_pJobSrcAddrPtr	Pointer to current position in source data buffer.
Mem_43_Eep_DataType *	Mem_43_Eep_pJobDataDestPtr	Pointer to current position in target data buffer.

6.1.2.2 struct Mem_43_Eep_SectorBatchType

Sector Batch Type.

Sector Batch data structure for group of identical sectors Note: burst sizes equal to normal sizes in case burst disabled

Definition at line 220 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.2.3 struct Mem_43_Eep_MemInstanceType

Mem Instance Type.

Mem Instance data structure

Definition at line 237 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.2.4 struct Mem_43_Eep_ConfigType

Mem_43_Eep Config Type.

Mem_43_Eep module initialization data structure

Definition at line 252 of file [Mem_43_Eep_InternalTypes.h](#).

Data Fields

- const [Mem_43_Eep_MemInstanceType](#) * [MemInstances](#)
Point to first element in array of mem instances configurations.
- [Mem_43_Eep_CrcType](#) [u16ConfigCrc](#)
MEM_43_EEP Config Set CRC checksum.
- const [Sd_Emmc_Ip_ConfigType](#) * [pMem_43_EepSdConfig](#)
SD Ip configuration pointer.

6.1.2.4.1 Field Documentation

6.1.2.4.1.1 MemInstances const [Mem_43_Eep_MemInstanceType](#)* MemInstances

Point to first element in array of mem instances configurations.

Definition at line 257 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.2.4.1.2 u16ConfigCrc [Mem_43_Eep_CrcType](#) u16ConfigCrc

MEM_43_EEP Config Set CRC checksum.

Definition at line 261 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.2.4.1.3 pMem_43_EepSdConfig `const Sd_Emmc_Ip_ConfigType* pMem_43_EepSdConfig`

SD Ip configuration pointer.

Definition at line 265 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.3 Macro Definition Documentation

6.1.3.1 MEM_43_EEP_E_UNINIT

```
#define MEM_43_EEP_E_UNINIT
```

Development error codes (passed to DET).

service ID of error: API service called without module initialization. (passed to DET)

Definition at line 131 of file [Mem_43_Eep.h](#).

6.1.3.2 MEM_43_EEP_E_PARAM_POINTER

```
#define MEM_43_EEP_E_PARAM_POINTER
```

service ID of error: NULL_PTR passed. (passed to DET)

Definition at line 135 of file [Mem_43_Eep.h](#).

6.1.3.3 MEM_43_EEP_E_PARAM_ADDRESS

```
#define MEM_43_EEP_E_PARAM_ADDRESS
```

service ID of error: TargetAddress is not in range and aligned to first byte of mem_43_eeprom sector. (passed to DET)

Definition at line 139 of file [Mem_43_Eep.h](#).

6.1.3.4 MEM_43_EEP_E_PARAM_LENGTH

```
#define MEM_43_EEP_E_PARAM_LENGTH
```

service ID of error: TargetAddress is not in range and aligned to last byte of mem_43_eeprom sector. (passed to DET)

Definition at line 143 of file [Mem_43_Eep.h](#).

6.1.3.5 MEM_43_EEP_E_PARAM_INSTANCE_ID

```
#define MEM_43_EEP_E_PARAM_INSTANCE_ID
```

service ID of error: invalid instace id passed. (passed to DET)

Definition at line 147 of file [Mem_43_Eep.h](#).

6.1.3.6 MEM_43_EEP_E_JOB_PENDING

```
#define MEM_43_EEP_E_JOB_PENDING
```

service ID of error: invalid instace id passed. (passed to DET)

Definition at line 151 of file [Mem_43_Eep.h](#).

6.1.3.7 MEM_43_EEP_E_TIMEOUT

```
#define MEM_43_EEP_E_TIMEOUT
```

service ID of error: The hardware operation did not finish before timeout expired. (passed to DET)

Definition at line 155 of file [Mem_43_Eep.h](#).

6.1.3.8 MEM_43_EEP_INIT_ID

```
#define MEM_43_EEP_INIT_ID
```

All service IDs (passed to DET).

Definition at line 161 of file [Mem_43_Eep.h](#).

6.1.3.9 MEM_43_EEP_START_SEC_CODE

```
#define MEM_43_EEP_START_SEC_CODE
```

Start of Mem_43_Eep section CODE.

Definition at line 184 of file [Mem_43_Eep.h](#).

6.1.3.10 MEM_43_EEP_STOP_SEC_CODE

```
#define MEM_43_EEP_STOP_SEC_CODE
```

Stop of Mem_43_Eep section CODE.

Definition at line 478 of file [Mem_43_Eep.h](#).

6.1.4 Types Reference

6.1.4.1 Mem_43_Eep_CrcType

```
typedef uint16 Mem_43_Eep_CrcType
```

Mem_43_Eep CRC Type.

CRC computed over config set.

Definition at line 82 of file [Mem_43_Eep_Types.h](#).

6.1.4.2 Mem_43_Eep_AddressType

```
typedef uint32 Mem_43_Eep_AddressType
```

Mem_43_Eep Address Type.

Physical memory device address type.

Definition at line 90 of file [Mem_43_Eep_Types.h](#).

6.1.4.3 Mem_43_Eep_DataType

```
typedef uint8 Mem_43_Eep_DataType
```

Mem_43_Eep Data Type.

Read data user buffer type.

Definition at line 98 of file [Mem_43_Eep_Types.h](#).

6.1.4.4 Mem_43_Eep_InstanceIdType

```
typedef uint32 Mem_43_Eep_InstanceIdType
```

Mem_43_Eep InstanceId Type.

Job end notification function called by Mem in case the job processing is configured for job end notification.

Definition at line 107 of file [Mem_43_Eep_Types.h](#).

6.1.4.5 Mem_43_Eep_LengthType

```
typedef uint32 Mem_43_Eep_LengthType
```

Mem_43_Eep Length Type.

Number of bytes to read,write,erase,compare

Definition at line 147 of file [Mem_43_Eep_Types.h](#).

6.1.4.6 Mem_43_Eep_HwServiceIdType

```
typedef uint32 Mem_43_Eep_HwServiceIdType
```

Mem_43_Eep HwServiceId Type.

Job end notification function called by Mem in case the job processing is configured for job end notification.

Definition at line 156 of file [Mem_43_Eep_Types.h](#).

6.1.5 Enum Reference

6.1.5.1 Mem_43_Eep_JobType

```
enum Mem_43_Eep_JobType
```

Type of job currently executed by Mem_43_Eep_MainFunction.

Enumerator

MEM_43_EEP_JOB_ERASE	erase one or more complete mem_eeprom sectors
MEM_43_EEP_JOB_WRITE	write one or more complete mem_eeprom pages
MEM_43_EEP_JOB_READ	read one or more bytes from mem_eeprom memory
MEM_43_EEP_JOB_BLANKCHECK	check blank bytes from mem_eeprom memory
MEM_43_EEP_JOB_HWSPECIFICSERVICE	hardware specific service

Definition at line 103 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.5.2 Mem_43_Eep_BlankCheckType

enum [Mem_43_Eep_BlankCheckType](#)

Asynchronous job result type.

Enumerator

MEM_43_EEP_BLANKCHECK_E_OK	Has been finished successfully.
MEM_43_EEP_BLANKCHECK_E_NOT_OK	Has not been finished successfully.
MEM_43_EEP_BLANKCHECK_E_INCONSISTENT	The checked page is not blank.

Definition at line 130 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.5.3 Mem_43_Eep_CompareCheckType

enum [Mem_43_Eep_CompareCheckType](#)

Result of compare operation.

Definition at line 149 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.5.4 Mem_43_Eep_CrcDataSizeType

enum [Mem_43_Eep_CrcDataSizeType](#)

Size of data to be processed by CRC.

Definition at line 161 of file [Mem_43_Eep_InternalTypes.h](#).

6.1.5.5 Mem_43_Eep_JobResultType

enum [Mem_43_Eep_JobResultType](#)

Asynchronous job result type.

Enumerator

MEM_43_EEP_JOB_OK	The last job has been finished successfully.
MEM_43_EEP_JOB_PENDING	A job is currently being processed.
MEM_43_EEP_JOB_FAILED	Job failed for some unspecific reason.
MEM_43_EEP_INCONSISTENT	The checked page is not blank.
MEM_43_EEP_ECC_UNCORRECTED	Uncorrectable ECC errors occurred during memory access.
MEM_43_EEP_ECC_CORRECTED	Correctable ECC errors occurred during memory access.

Definition at line 113 of file [Mem_43_Eep_Types.h](#).

6.1.6 Function Reference

6.1.6.1 Mem_43_Eep_Init()

```
void Mem_43_Eep_Init (
    const Mem_43_Eep_ConfigType * ConfigPtr )
```

The function initializes Mem_43_Eep module.

The function sets the internal module variables according to given configuration set.

Parameters

in	<i>ConfigPtr</i>	Pointer to mem_43_eeprom driver configuration set.
----	------------------	--

Precondition

ConfigPtr must not be NULL_PTR

6.1.6.2 Mem_43_Eep_DeInit()

```
void Mem_43_Eep_DeInit (
    void )
```

The function deinitializes Mem_43_Eep module.

The function sets the internal module variables to null.

Parameters

in	<i>None</i>	
----	-------------	--

6.1.6.3 Mem_43_Eep_MainFunction()

```
void Mem_43_Eep_MainFunction (
    void )
```

Service to handle the requested jobs and the internal management operations.

6.2 SD_EMMC

6.2.1 Detailed Description

Data Structures

- struct [Sd_Ip_CidType](#)
SD card CID register. [More...](#)
- struct [Sd_Ip_MmcCidType](#)
MMC card CID register. [More...](#)
- struct [Sd_Ip_CsdType](#)
SD card CSD register. [More...](#)
- struct [Sd_Ip_ScrType](#)
SD card SCR register. [More...](#)
- struct [Sd_Ip_MmcCsdType](#)
MMC and MMC card CSD register. [More...](#)
- struct [Sd_Ip_MmcECsrType](#)
MMC and MMC card Extended CSD register (unit: byte). [More...](#)
- struct [Usdhc_Ip_TransferStorageType](#)
SDHC transfer storage. [More...](#)
- struct [Sd_Emmc_Ip_BootConfigType](#)
SD and MMC card state. [More...](#)
- struct [Sd_Ip_MmcCapabilityType](#)
EMMC card capability information. [More...](#)
- struct [Sd_Emmc_Ip_CardInformationType](#)
SD and MMC card information. [More...](#)
- struct [Sd_Emmc_Ip_ConfigType](#)
SD and MMC card state. [More...](#)

Macros

- `#define SDMMC_CLOCK_100KHZ`
SD/MMC card initialization clock frequency.
- `#define SDMMC_CLOCK_400KHZ`
SD/MMC card initialization clock frequency.
- `#define SD_CLOCK_25MHZ`
SD card bus frequency 1 in high speed mode.
- `#define SD_CLOCK_26MHZ`
SD card bus frequency 1 in high speed mode.
- `#define SD_CLOCK_40MHZ`
SD card bus frequency 2 in high speed mode.
- `#define SD_CLOCK_50MHZ`
SD card bus frequency 3 in high speed mode.
- `#define MMC_CLOCK_26MHZ`
MMC card bus frequency 1 in high speed mode.
- `#define MMC_CLOCK_52MHZ`
MMC card bus frequency 2 in high speed mode.
- `#define MMC_CLOCK_200MHZ`
MMC card bus frequency 3 in HS200 or HS400 mode.
- `#define MMCSetBits`
Access Set Bits field.
- `#define MMCClearBits`
Access Clear Bits field.
- `#define MMCCWriteBits`
Access Write Bits field.
- `#define MMCCCommandSets`
Access Command Set field.
- `#define MMCShiftIndex`
Shift index field.
- `#define MMCDefaultSpeed`
Default Speed field.
- `#define MMCHightSpeed`
High Speed field.
- `#define MMCHS200Speed`
HS200 Speed field.
- `#define MMCHS400Speed`
HS400 Speed field.
- `#define MMCPowerClass100`
Power Class 100 field.
- `#define MMCPowerClass120`
Power Class 120 field.
- `#define MMCPowerClass150`
Power Class 150 field.
- `#define MMCPowerClass180`
Power Class 180 field.
- `#define MMCPowerClass200`

- Power Class 200 field.*

 - #define [MMCPowerClass220](#)
- Power Class 220 field.*

 - #define [MMCPowerClass250](#)
- Power Class 250 field.*

 - #define [MMCPowerClass300](#)
- Power Class 300 field.*

 - #define [MMCPowerClass350](#)
- Power Class 350 field.*

 - #define [MMCPowerClass400](#)
- Power Class 400 field.*

 - #define [MMCPowerClass450](#)
- Power Class 450 field.*

 - #define [MMCPowerClass500](#)
- Power Class 500 field.*

 - #define [MMCPowerClass600](#)
- Power Class 600 field.*

 - #define [MMCPowerClass700](#)
- Power Class 700 field.*

 - #define [MMCPowerClass800](#)
- Power Class 800 field.*

 - #define [MMCPowerClass900](#)
- Power Class >800 field.*

 - #define [MMCBusWidth1](#)
- Bus width 1 bit field.*

 - #define [MMCBusWidth4](#)
- Bus width 4 bits field.*

 - #define [MMCBusWidth8](#)
- Bus width 8 bits field.*

 - #define [MMCBusWidth4ddr](#)
- Bus width 4 bits ddr field.*

 - #define [MMCBusWidth8ddr](#)
- Bus width 8 bit ddr field.*

 - #define [MMC_ALTERNATE_BOOT_SUPPORT_MASK](#)
- Alternate boot support(BOOT_INFO in Extended CSD)*

 - #define [MMC_POWER_CLASS_4BIT_MASK](#)
- The power class value bit mask when bus in 4 bit mode.*

 - #define [MMC_POWER_CLASS_8BIT_MASK](#)
- The power class current value bit mask when bus in 8 bit mode.*

 - #define [MMC_PARTITION_CONFIG_ACCESS_SHIFT](#)
- The bit shift for PARTITION ACCESS filed in PARTITION_CONFIG (PARTITION_CONFIG in Extend CSD)*

 - #define [MMC_PARTITION_CONFIG_ACCESS_MASK](#)
- The bit mask for PARTITION ACCESS field in PARTITION_CONFIG.*

 - #define [MMC_PARTITION_CONFIG_ACCESS\(x\)](#)
- The bit set for PARTITION ACCESS field in PARTITION_CONFIG.*

 - #define [MMC_PARTITION_CONFIG_ENABLE_SHIFT](#)
- The bit shift for PARTITION ENABLE field in PARTITION_CONFIG.*

- `#define MMC_PARTITION_CONFIG_ENABLE_MASK`
The bit mask for PARTITION ENABLE field in PARTITION_CONFIG.
- `#define MMC_PARTITION_CONFIG_ENABLE(x)`
The bit set for PARTITION ENABLE field in PARTITION_CONFIG.
- `#define MMC_PARTITION_CONFIG_ACK_SHIFT`
The bit shift for ACK field in PARTITION_CONFIG.
- `#define MMC_PARTITION_CONFIG_ACK_MASK`
The bit mask for ACK field in PARTITION_CONFIG.
- `#define MMC_PARTITION_CONFIG_ACK(x)`
The bit set for ACK field in PARTITION_CONFIG.
- `#define MMC_BOOT_BUS_WIDTH_WIDTH_SHIFT`
The bit shift for BOOT BUS WIDTH field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_BUS_WIDTH_WIDTH_MASK`
The bit mask for BOOT BUS WIDTH field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_BUS_WIDTH_WIDTH(x)`
The bit set for BOOT BUS WIDTH field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_BUS_WIDTH_RESET_SHIFT`
The bit shift for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_BUS_WIDTH_RESET_MASK`
The bit mask for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_BUS_WIDTH_RESET(x)`
The bit set for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_MODE_SHIFT`
The bit shift for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_MODE_MASK`
The bit mask for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.
- `#define MMC_BOOT_MODE(x)`
The bit set for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.
- `#define MMC_HS_BOOT_MODE_SHIFT`
The bit shift for HS_BOOT_MODE field in BOOT_INFO.
- `#define MMC_HS_BOOT_MODE_MASK`
The bit mask for HS_BOOT_MODE field in BOOT_INFO.
- `#define MMC_HS_BOOT_MODE(x)`
The bit set for HS_BOOT_MODE field in BOOT_INFO.
- `#define MMC_DDR_BOOT_MODE_SHIFT`
The bit shift for DDR_BOOT_MODE field in BOOT_INFO.
- `#define MMC_DDR_BOOT_MODE_MASK`
The bit mask for DDR_BOOT_MODE field in BOOT_INFO.
- `#define MMC_DDR_BOOT_MODE(x)`
The bit set for DDR_BOOT_MODE field in BOOT_INFO.
- `#define SD_IP_R1_OUT_OF_RANGE_ERROR`
Card status bit in R1.
- `#define SD_IP_R1_ADDRESS_ERROR`
- `#define SD_IP_R1_BLOCK_LENGTH_ERROR`
- `#define SD_IP_R1_ERASE_SEQ_ERROR`
- `#define SD_IP_R1_ERASE_PARAMETER_ERROR`
- `#define SD_IP_R1_WRITE_PROTECT_VIOLATION`

- #define SD_IP_R1_CARD_IS_LOCKED
- #define SD_IP_R1_LOCK_UNLOCK_FAILED
- #define SD_IP_R1_COMMAND_CRC_FAILED
- #define SD_IP_R1_ILLEGAL_COMMAND
- #define SD_IP_R1_CARD_ECC_FAILED
- #define SD_IP_R1_CARD_CONTROLLER_ERROR
- #define SD_IP_R1_UNKNOWN_ERROR
- #define SD_IP_R1_CID_CSD_OVERWRITE
- #define SD_IP_R1_WRITE_PROTECT_ERASE_SKIP_FLAG
- #define SD_IP_R1_CARD_ECC_DISABLED_FLAG
- #define SD_IP_R1_ERASE_RESET_FLAG
- #define SD_IP_R1_READY_FOR_DATA_FLAG
- #define SD_IP_R1_SWITCH_ERROR_FLAG
- #define SD_IP_R1_APPLICATIONC_COMMAND_FLAG
- #define SD_IP_R1_AUTHENTICATE_SEQ_ERROR
- #define SD_IP_R1_CURRENT_STATE(x)
R1: current state.
- #define aSD_OcrPowerUpBusyFlag
OCR register in SD card.
- #define aSD_OcrHostCapacitySupportFlag
- #define aSD_OcrCardCapacitySupportFlag
- #define aSD_OcrSwitch18RequestFlag
- #define aSD_OcrSwitch18AcceptFlag
- #define aSD_OcrSwitchSDXCPowerControlFlag
- #define aSD_OcrVdd27_28Flag
- #define aSD_OcrVdd28_29Flag
- #define aSD_OcrVdd29_30Flag
- #define aSD_OcrVdd30_31Flag
- #define aSD_OcrVdd31_32Flag
- #define aSD_OcrVdd32_33Flag
- #define aSD_OcrVdd33_34Flag
- #define aSD_OcrVdd34_35Flag
- #define aSD_OcrVdd35_36Flag
- #define SD_IP_CSD_READ_BLOCK_PARTIAL_FLAG
SD card CSD register flags.
- #define SD_IP_CSD_WRITE_BLOCK_MISALIGN_FLAG
- #define SD_IP_CSD_READ_BLOCK_MISALIGN_FLAG
- #define SD_IP_CSD_DSR_IMPLEMENTATION_FLAG
- #define SD_IP_CSD_ERASE_BLOCK_ENABLED_FLAG
- #define SD_IP_CSD_WRITE_PROTECT_GROUP_ENABLED_FLAG
- #define SD_IP_CSD_WRITE_BLOCK_PARTIAL_FLAG
- #define SD_IP_CSD_FILE_FORMAT_GROUP_FLAG
- #define SD_IP_CSD_COPY_FLAG
- #define SD_IP_CSD_PERMANENT_WRITE_PROTECT_FLAG
- #define SD_IP_CSD_TEMPORARY_WRITE_PROTECT_FLAG
- #define MMC_OCR_V170TO195_SHIFT
The bit mask for VOLTAGE WINDOW 1.70V to 1.95V field in OCR.
- #define MMC_OCR_V170TO195_MASK
The bit mask for VOLTAGE WINDOW 1.70V to 1.95V field in OCR.
- #define MMC_OCR_V200TO260_SHIFT

- The bit shift for VOLTAGE WINDOW 2.00V to 2.60V field in OCR.*

 - #define [MMC_OCR_V200TO260_MASK](#)
- The bit mask for VOLTAGE WINDOW 2.00V to 2.60V field in OCR.*

 - #define [MMC_OCR_V270TO360_SHIFT](#)
- The bit shift for VOLTAGE WINDOW 2.70V to 3.60V field in OCR.*

 - #define [MMC_OCR_V270TO360_MASK](#)
- The bit mask for VOLTAGE WINDOW 2.70V to 3.60V field in OCR.*

 - #define [MMC_OCR_ACCESS_MODE_SHIFT](#)
- The bit shift for ACCESS MODE field in OCR.*

 - #define [MMC_OCR_ACCESS_MODE_MASK](#)
- The bit mask for ACCESS MODE field in OCR.*

 - #define [MMC_OCR_SECTOR_ACCESS_MODE_MASK](#)
- The bit mask for SECTOR ACCESS MODE in OCR.*

 - #define [MMC_OCR_BUSY_SHIFT](#)
- The bit shift for BUSY field in OCR.*

 - #define [MMC_OCR_BUSY_MASK](#)
- The bit mask for BUSY field in OCR.*

 - #define [MMC_TRANSFER_FREQUENCY_UNIT_SHIFT](#)
- The bit shift for FREQUENCY UNIT field in TRANSFER SPEED(TRAN-SPEED in Extended CSD)*

 - #define [MMC_TRANSFER__FREQUENCY_UNIT_MASK](#)
- The bit mask for FRQEQUENCY UNIT in TRANSFER SPEED.*

 - #define [MMC_TRANSFER_MULTIPLIER_SHIFT](#)
- The bit shift for MULTIPLIER field in TRANSFER SPEED.*

 - #define [MMC_TRANSFER_MULTIPLIER_MASK](#)
- The bit mask for MULTIPLIER field in TRANSFER SPEED*
- #define [READ_MMC_TRANSFER_SPEED_FREQUENCY_UNIT](#)(CSD)

Read the value of FREQUENCY UNIT in TRANSFER SPEED.
- #define [READ_MMC_TRANSFER_SPEED_MULTIPLIER](#)(CSD)

Read the value of MULTIPLIER filed in TRANSFER SPEED.
- #define [MMC_EXT_CSD_SEC_CNT_SHIFT](#)

The bit shift for MMC_EXT_CSD_SEC_CNT field in extended CSD register.
- #define [MMC_EXT_CSD_DATA_SECTOR_SIZE_SHIFT](#)

The bit shift for MMC_EXT_CSD_DATA_SECTOR_SIZE field in extended CSD register.
- #define [MMC_EXT_CSD_BUS_WIDTH_SHIFT](#)

The bit shift for MMC_EXT_CSD_BUS_WIDTH field in extended CSD register.
- #define [MMC_EXT_CSD_REV_SHIFT](#)

The bit shift for MMC_EXT_CSD_REV field in extended CSD register.
- #define [MMC_EXT_CSD_STRUCTURE_SHIFT](#)

The bit shift for MMC_EXT_CSD_STRUCTURE field in extended CSD register.
- #define [MMC_EXT_CSD_CARD_TYPE_SHIFT](#)

The bit shift for MMC_EXT_CSD_CARD_TYPE field in extended CSD register.
- #define [MMC_EXT_CSD_CARD_TYPE_MASK](#)

The bit mask for MMC_EXT_CSD_CARD_TYPE field in extended CSD register.
- #define [MMC_EXT_CSD_STROBE_SUPPORT_SHIFT](#)

The bit shift for MMC_EXT_CSD_STROBE_SUPPORT field in extended CSD register.
- #define [MMC_EXT_CSD_HS_TIMING_SHIFT](#)

- The bit shift for MMC_EXT_CSD_STROBE_SUPPORT field in extended CSD register.*

 - #define `MMC_EXT_CSD_PARTITION_CONFIG_SHIFT`
- The bit shift for MMC_EXT_CSD_INDEX_PARTITION_CONFIG field in extended CSD register.*

 - #define `MMC_EXT_CSD_BOOT_SIZE_MULT_SHIFT`
- The bit shift for MMC_EXT_CSD_INDEX_BOOT_SIZE_MULT field in extended CSD register.*

 - #define `MMC_EXT_CSD_BOOT_INFO_SHIFT`
- The bit shift for MMC_EXT_CSD_INDEX_BOOT_INFO field in extended CSD register.*

 - #define `MMC_EXT_CSD_BOOT_BUS_CONDITIONS_SHIFT`
- The bit shift for MMC_EXT_CSD_INDEX_BOOT_BUS_CONDITIONS field in extended CSD register.*

 - #define `MMC_EXT_CSD_INDEX_FLUSH_CACHE`
- EXT CSD byte index.*

 - #define `MMC_EXT_CSD_INDEX_CACHE_CTRL`
 - #define `MMC_EXT_CSD_INDEX_POWER_OFF_NOTIFICATION`
 - #define `MMC_EXT_CSD_INDEX_EXP_EVENTS_STATUS`
 - #define `MMC_EXT_CSD_INDEX_DATA_SECTOR_SIZE`
 - #define `MMC_EXT_CSD_INDEX_NATIVE_SECTOR_SIZE`
 - #define `MMC_EXT_CSD_INDEX_GP_SIZE_MULT`
 - #define `MMC_EXT_CSD_INDEX_PARTITION_ATTRIBUTE`
 - #define `MMC_EXT_CSD_INDEX_PARTITION_SUPPORT`
 - #define `MMC_EXT_CSD_INDEX_HPI_MGMT`
 - #define `MMC_EXT_CSD_INDEX_RST_N_FUNCTION`
 - #define `MMC_EXT_CSD_INDEX_BKOPS_EN`
 - #define `MMC_EXT_CSD_INDEX_BKOPS_START`
 - #define `MMC_EXT_CSD_INDEX_SANITIZE_START`
 - #define `MMC_EXT_CSD_INDEX_WR_REL_PARAM`
 - #define `MMC_EXT_CSD_INDEX_USER_WP`
 - #define `MMC_EXT_CSD_INDEX_BOOT_WP`
 - #define `MMC_EXT_CSD_INDEX_BOOT_WP_STATUS`
 - #define `MMC_EXT_CSD_INDEX_ERASE_GROUP_DEF`
 - #define `MMC_EXT_CSD_INDEX_BOOT_BUS_CONDITIONS`
 - #define `MMC_EXT_CSD_INDEX_BOOT_CONFIG_PROT`
 - #define `MMC_EXT_CSD_INDEX_PARTITION_CONFIG`
 - #define `MMC_EXT_CSD_INDEX_ERASED_MEM_CONT`
 - #define `MMC_EXT_CSD_INDEX_BUS_WIDTH`
 - #define `MMC_EXT_CSD_STROBE_SUPPORT`
 - #define `MMC_EXT_CSD_INDEX_HS_TIMING`
 - #define `MMC_EXT_CSD_INDEX_POWER_CLASS`
 - #define `MMC_EXT_CSD_INDEX_REV`
 - #define `MMC_EXT_CSD_INDEX_STRUCTURE`
 - #define `MMC_EXT_CSD_INDEX_CARD_TYPE`
 - #define `MMC_EXT_CSD_INDEX_OUT_OF_INTERRUPT_TIME`
 - #define `MMC_EXT_CSD_INDEX_PART_SWITCH_TIME`
 - #define `MMC_EXT_CSD_INDEX_PWR_CL_52_195`
 - #define `MMC_EXT_CSD_INDEX_PWR_CL_26_195`
 - #define `MMC_EXT_CSD_INDEX_PWR_CL_52_360`
 - #define `MMC_EXT_CSD_INDEX_PWR_CL_26_360`
 - #define `MMC_EXT_CSD_INDEX_SEC_CNT`
 - #define `MMC_EXT_CSD_INDEX_S_A_TIMEOUT`
 - #define `MMC_EXT_CSD_INDEX_REL_WR_SEC_C`
 - #define `MMC_EXT_CSD_INDEX_HC_WP_GRP_SIZE`

- `#define MMC_EXT_CSD_INDEX_ERASE_TIMEOUT_MULT`
- `#define MMC_EXT_CSD_INDEX_HC_ERASE_GRP_SIZE`
- `#define MMC_EXT_CSD_INDEX_BOOT_SIZE_MULT`
- `#define MMC_EXT_CSD_INDEX_BOOT_INFO`
- `#define MMC_EXT_CSD_INDEX_SEC_TRIM_MULT`
- `#define MMC_EXT_CSD_INDEX_SEC_ERASE_MULT`
- `#define MMC_EXT_CSD_INDEX_SEC_FEATURE_SUPPORT`
- `#define MMC_EXT_CSD_INDEX_TRIM_MULT`
- `#define MMC_EXT_CSD_INDEX_PWR_CL_200_130`
- `#define MMC_EXT_CSD_INDEX_PWR_CL_200_195`
- `#define MMC_EXT_CSD_INDEX_PWR_CL_DDR_52_195`
- `#define MMC_EXT_CSD_INDEX_PWR_CL_DDR_52_360`
- `#define MMC_EXT_CSD_INDEX_BKOPS_STATUS`
- `#define MMC_EXT_CSD_INDEX_POWER_OFF_LONG_TIME`
- `#define MMC_EXT_CSD_INDEX_GENERIC_CMD6_TIME`
- `#define MMC_EXT_CSD_INDEX_CACHE_SIZE`
- `#define MMC_EXT_CSD_INDEX_TAG_UNIT_SIZE`
- `#define MMC_EXT_CSD_INDEX_DATA_TAG_SUPPORT`
- `#define MMC_EXT_CSD_INDEX_MAX_PACKED_WRITES`
- `#define MMC_EXT_CSD_INDEX_MAX_PACKED_READS`
- `#define MMC_EXT_CSD_INDEX_BKOPS_SUPPORT`
- `#define MMC_EXT_CSD_INDEX_HPI_FEATURES`
- `#define MMC_EXTENDED_CSD_BYTES`
The length of Extended CSD register, unit as bytes.
- `#define MMC_DEFAULT_RELATIVE_ADDRESS`
MMC card default relative address.
- `#define SD_PRODUCT_NAME_BYTES`
SD card product name length united as bytes.
- `#define MMC_PRODUCT_NAME_BYTES`
MMC card product name length united as bytes.
- `#define SD_TRANSFER_SPEED_RATE_UNIT_SHIFT`
The bit shift for RATE UNIT field in TRANSFER SPEED.
- `#define SD_TRANSFER_SPEED_RATE_UNIT_MASK`
The bit mask for RATE UNIT field in TRANSFER SPEED.
- `#define SD_TRANSFER_SPEED_TIME_VALUE_SHIFT`
The bit shift for TIME VALUE field in TRANSFER SPEED.
- `#define SD_TRANSFER_SPEED_TIME_VALUE_MASK`
The bit mask for TIME VALUE field in TRANSFER SPEED.
- `#define SD_RD_TRANSFER_SPEED_RATE_UNIT(x)`
Read the value of FREQUENCY UNIT in TRANSFER SPEED field.
- `#define SD_RD_TRANSFER_SPEED_TIME_VALUE(x)`
Read the value of TIME VALUE in TRANSFER SPEED field.
- `#define MMC_SWITCH_COMMAND_SET_SHIFT`
The bit shift for COMMAND SET field in SWITCH command.
- `#define MMC_SWITCH_COMMAND_SET_MASK`
The bit mask for COMMAND set field in SWITCH command.
- `#define MMC_SWITCH_VALUE_SHIFT`
The bit shift for VALUE field in SWITCH command.

- `#define MMC_SWITCH_VALUE_MASK`
The bit mask for VALUE field in SWITCH command.
- `#define MMC_SWITCH_BYTE_INDEX_SHIFT`
The bit shift for BYTE INDEX field in SWITCH command.
- `#define MMC_SWITCH_BYTE_INDEX_MASK`
The bit mask for BYTE INDEX field in SWITCH command.
- `#define MMC_SWITCH_ACCESS_MODE_SHIFT`
The bit shift for ACCESS MODE field in SWITCH command.
- `#define MMC_SWITCH_ACCESS_MODE_MASK`
The bit mask for ACCESS MODE field in SWITCH command.

Types Reference

- `typedef uint8 Sd_Ip_ProtocolType`
SD or MMC card type: memory card type that can be plugged into the SD port.

Enum Reference

- `enum Sd_Emmc_Ip_StatusType`
SD/MMC generic status.
- `enum Sd_Ip_R1CurrentStateType`
CURRENT_STATE field in R1.
- `enum Sd_Ip_CommandType`
SD card individual commands.
- `enum Sd_Ip_ApplicationCmdType`
SD card individual application commands.
- `enum Sd_Ip_SDMMC_CommandType`
SD/MMC card common commands.
- `enum Sd_Ip_MmcCommandType`
MMC card individual commands.
- `enum Sd_Ip_CommandClassType`
SD card command class.
- `enum Sd_Ip_SpecificationVersionType`
SD card specification version number.
- `enum Sd_Ip_DataBusWidthType`
SD card bus width.
- `enum Sd_Ip_SwitchModeType`
SD card switch mode.
- `enum Sd_Ip_TimingFunctionType`
SD timing function number.
- `enum Sd_Ip_GroupNumType`
SD group number.
- `enum Sd_Ip_DriverStrenghtType`
SD card driver strength.
- `enum Sd_Ip_MaxCurrentType`

- *SD card current limit.*
- enum [Sd_Ip_ScrFlagType](#)
SD card SCR register flags.
- enum [Sd_Ip_MmcECsdFlagType](#)
MMC card CSD register flags.
- enum [Sd_Ip_CardFlagType](#)
Memory card flags.
- enum [Sd_Ip_MmcPartitionAccessType](#)
MMC card partition access selection.
- enum [Sd_Ip_MmcBootPartitionType](#)
MMC card boot partition selection.
- enum [Sd_Ip_MmcBootSpeedType](#)
MMC card boot mode selection.
- enum [Sd_Ip_MmcBootWidthType](#)
MMC card boot width selection.
- enum [Sd_Ip_MmcBusWidthType](#)
MMC card bus width selection.
- enum [Sd_Ip_MmcCardType](#)
MMC card speed supports.
- enum [Sd_Ip_MmcSpeedModeType](#)
MMC card speed supports.

6.2.2 Data Structure Documentation

6.2.2.1 struct Sd_Ip_CidType

SD card CID register.

Definition at line 647 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	manufacturerID	Manufacturer ID [127:120]
uint16	applicationID	OEM/Application ID [119:104]
uint8	productName[(5U)]	Product name [103:64]
uint8	productVersion	Product revision [63:56]
uint32	productSerialNumber	Product serial number [55:24]
uint16	manufacturerData	Manufacturing date [19:8]

6.2.2.2 struct Sd_Ip_MmcCidType

MMC card CID register.

Module Documentation

Definition at line 658 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint32	productSerialNumber	Product serial number
uint16	applicationID	OEM/Application ID
uint8	manufacturerID	Manufacturer ID
uint8	productName[(6U)]	Product name
uint8	productVersion	Product revision
uint8	manufacturerData	Manufacturing date

6.2.2.3 struct Sd_Ip_CsdType

SD card CSD register.

Definition at line 669 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	csdStructure	CSD structure [127:126]
uint8	dataReadAccessTime1	Data read access-time-1 [119:112]
uint8	dataReadAccessTime2	Data read access-time-2 in clock cycles (NSAC*100) [111:104]
uint8	transferSpeed	Maximum data transfer rate [103:96]
uint16	cardCommandClass	Card command classes [95:84]
uint8	readBlockLength	Maximum read data block length [83:80]
uint16	flags	Flags in sd_csd_flag_t
uint32	deviceSize	Device size [73:62]
uint8	readCurrentVddMin	Maximum read current at VDD min [61:59]
uint8	readCurrentVddMax	Maximum read current at VDD max [58:56]
uint8	writeCurrentVddMin	Maximum write current at VDD min [55:53]
uint8	writeCurrentVddMax	Maximum write current at VDD max [52:50]
uint8	deviceSizeMultiplier	Device size multiplier [49:47]
uint8	eraseSectorSize	Erase sector size [45:39]
uint8	writeProtectGroupSize	Write protect group size [38:32]
uint8	writeSpeedFactor	Write speed factor [28:26]
uint8	writeBlockLength	Maximum write data block length [25:22]
uint8	fileFormat	File format [11:10]

6.2.2.4 struct Sd_Ip_ScrType

SD card SCR register.

Definition at line 709 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	scrStructure	SCR Structure [63:60]
uint8	sdSpecification	SD memory card specification version [59:56]
uint16	flags	SCR flags in Sd_Ip_ScrFlagType
uint8	sdSecurity	Security specification supported [54:52]
uint8	sdBusWidths	Data bus widths supported [51:48]
uint8	extendedSecurity	Extended security support [46:43]
uint8	commandSupport	Command support bits [33:32]
uint32	reservedForManufacturer	reserved for manufacturer usage [31:0]

6.2.2.5 struct Sd_Ip_MmcCsdType

MMC and MMC card CSD register.

Definition at line 722 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	csdStructureVersion	CSD structure [127:126]
uint8	systemSpecificationVersion	System specification version [125:122]
uint8	dataReadAccessTime1	Data read access-time 1 [119:112]
uint8	dataReadAccessTime2	Data read access-time 2 in CLOCK cycles (NSAC*100) [111:104]
uint8	transferSpeed	Max. bus clock frequency [103:96]
uint16	cardCommandClass	card command classes [95:84]
uint8	readBlockLength	Max. read data block length [83:80]
uint16	flags	Contain flags in Sd_Ip_MmcECsdFlagType
uint16	deviceSize	Device size [73:62]
uint8	readCurrentVddMin	Max. read current @ VDD min [61:59]
uint8	readCurrentVddMax	Max. read current @ VDD max [58:56]
uint8	writeCurrentVddMin	Max. write current @ VDD min [55:53]
uint8	writeCurrentVddMax	Max. write current @ VDD max [52:50]
uint8	deviceSizeMultiplier	Device size multiplier [49:47]
uint8	eraseGroupSize	Erase group size [46:42]
uint8	eraseGroupSizeMultiplier	Erase group size multiplier [41:37]
uint8	writeProtectGroupSize	Write protect group size [36:32]
uint8	defaultEcc	Manufacturer default ECC [30:29]
uint8	writeSpeedFactor	Write speed factor [28:26]

Data Fields

Type	Name	Description
uint8	maxWriteBlockLength	Max. write data block length [25:22]
uint8	fileFormat	File format [11:10]
uint8	eccCode	ECC code [9:8]

6.2.2.6 struct Sd_Ip_MmcECsrType

MMC and MMC card Extended CSD register (unit: byte).

Definition at line 749 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	highDensityEraseGroupDefinition	High-density erase group definition [175]
uint8	bootBusCondition	Boot bus condition [177]
uint8	bootConfig	Boot configuration [179]
uint8	eraseMemoryContent	Erased memory content [181]
uint8	dataBusWidth	Data bus width mode [183]
uint8	strobeSupport	Enhanced strobe support [184]
uint8	highSpeedTiming	High-speed interface timing [185]
uint8	powerClass	Power class [187]
uint8	commandSetRevision	Command set revision [189]
uint8	commandSet	Command set [191]
uint8	extendedCsdVersion	Extended CSD revision [192]
uint8	csdStructureVersion	CSD structure version [194]
uint8	cardType	Card Type [196]
uint8	powerClass52MHz195V	Power Class for 52MHz @ 1.95V [200]
uint8	powerClass26MHz195V	Power Class for 26MHz @ 1.95V [201]
uint8	powerClass52MHz360V	Power Class for 52MHz @ 3.6V [202]
uint8	powerClass26MHz360V	Power Class for 26MHz @ 3.6V [203]
uint8	minimumReadPerformance4Bit26MHz	Minimum Read Performance for 4bit at 26MHz [205]
uint8	minimumWritePerformance4Bit26MHz	Minimum Write Performance for 4bit at 26MHz [206]
uint8	minimumReadPerformance8Bit26MHz4Bit52MHz	Minimum read Performance for 8bit at 26MHz/4bit @52MHz [207]
uint8	minimumWritePerformance8Bit26MHz4Bit52MHz	Minimum Write Performance for 8bit at 26MHz/4bit @52MHz [208]
uint8	minimumReadPerformance8Bit52MHz	Minimum Read Performance for 8bit at 52MHz [209]

Data Fields

Type	Name	Description
uint8	minimumWritePerformance8Bit52MHz	Minimum Write Performance for 8bit at 52MHz [210]
uint32	sectorCount	Sector Count [215:212]
uint8	sleepAwakeTimeout	Sleep/awake timeout [217]
uint8	sleepCurrentVCCQ	Sleep current (VCCQ) [219]
uint8	sleepCurrentVCC	Sleep current (VCC) [220]
uint8	highCapacityWriteProtectGroupSize	High-capacity write protect group size [221]
uint8	reliableWriteSectorCount	Reliable write sector count [222]
uint8	highCapacityEraseTimeout	High-capacity erase timeout [223]
uint8	highCapacityEraseUnitSize	High-capacity erase unit size [224]
uint8	accessSize	Access size [225]
uint8	bootSizeMultiplier	Boot partition size [226]
uint8	bootInformation	Boot information [228]
uint8	supportedCommandSet	Supported Command Sets [504]
uint8	hdEraseGroupDef	High-density erase group definition [175]
uint8	dataSectorSize	Sector Size [61]
uint8	nativeSectorSize	Native sector Size [63]

6.2.2.7 struct Usdhc_Ip_TransferStorageType

SDHC transfer storage.

Definition at line 835 of file [Sd_Emmc_Ip_Types.h](#).

6.2.2.8 struct Sd_Emmc_Ip_BootConfigType

SD and MMC card state.

Define the card structure including the necessary fields to identify and describe the card.

Definition at line 893 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
Sd_Ip_MmcBootPartitionType	bootPartitionSelect	Boot partition selection
boolean	ackEnable	ACK enable/disabled for boot process
Sd_Ip_MmcBootSpeedType	bootSpeed	Boot speed - SDR or DDR
Sd_Ip_MmcBootWidthType	bootWidth	Boot width - 1-4-8 bits

6.2.2.9 struct Sd_Ip_MmcCapabilityType

EMMC card capability information.

Define structure to save the capability information of EMMC card.

Definition at line 938 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
boolean	supportHS400	Capability flags to indicate the support for HS400 mode
boolean	supportHS200	Capability flags to indicate the support for HS200 mode
boolean	supportHS52DDR	Capability flags to indicate the support for HS52DDR mode
boolean	supportHSLegacy	Capability flags to indicate the support for HS legacy mode
boolean	supportEnhanceStrobe	Capability flags to indicate the support for Enhanced strobe mode in HS400

6.2.2.10 struct Sd_Emmc_Ip_CardInformationType

SD and MMC card information.

Get all information of Sd Card/Emmc

Definition at line 952 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
Sd_Ip_MmcCidType *	EmmcCidRegister	MMC Extended CID Register
Sd_Ip_MmcCsdType *	EmmcCsdRegister	MMC Extended CSD Register
Sd_Ip_MmcECsrType *	EmmcCsrRegister	MMC Extended CSR Register
Sd_Ip_CidType *	SdCardCidRegister	Sd CID Register. Uniquely identifies the card.
Sd_Ip_CsdType *	SdCardCsdRegister	Sd CSD Register. Information about card operation conditions.
Sd_Ip_ScrType *	SdCardScrRegister	Sd SCR Register. Information about card's special features.
uint32	u32BlockCount	Card total block number
uint32	u32BlockSize	Card block size
uint32	u32BusClock_Hz	SD bus clock frequency united in Hz
uint32	u32Version	Card version - SPEC_VERS in the CSD
Sd_Ip_ProtocolType	eCardType	Card type

6.2.2.11 struct Sd_Emmc_Ip_ConfigType

SD and MMC card state.

Module Documentation

Define the card structure including the necessary fields to identify and describe the card.

Definition at line 972 of file [Sd_Emmc_Ip_Types.h](#).

Data Fields

Type	Name	Description
const Usdhc_Ip_ConfigType *	usdhcConfig	Host information
uint32	SdConfigClock	afrequencyTable
uint32	u32McuClock	Clock given by the Mcu to the Usdhc peripheral in Hz
uint32	flags	Flags in Sd_Ip_CardFlagType
Sd_Ip_ProtocolType	cardType	Card type
Sd_Ip_MmcBusWidthType	busWidthMode	Bus width - Only selectable in MMC mode
uint8	driverStrength	Card driver strength - Only apply for eMMC HS200 or HS400 mode
const Sd_Emmc_Ip_BootConfigType *	mmcBootConfig	Boot Configuration for MMC boot

6.2.3 Macro Definition Documentation

6.2.3.1 SDMMC_CLOCK_100KHZ

```
#define SDMMC_CLOCK_100KHZ
```

SD/MMC card initialization clock frequency.

Definition at line 112 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.2 SDMMC_CLOCK_400KHZ

```
#define SDMMC_CLOCK_400KHZ
```

SD/MMC card initialization clock frequency.

Definition at line 114 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.3 SD_CLOCK_25MHZ

```
#define SD_CLOCK_25MHZ
```

SD card bus frequency 1 in high speed mode.

Definition at line 116 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.4 SD_CLOCK_26MHZ

```
#define SD_CLOCK_26MHZ
```

SD card bus frequency 1 in high speed mode.

Definition at line 118 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.5 SD_CLOCK_40MHZ

```
#define SD_CLOCK_40MHZ
```

SD card bus frequency 2 in high speed mode.

Definition at line 120 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.6 SD_CLOCK_50MHZ

```
#define SD_CLOCK_50MHZ
```

SD card bus frequency 3 in high speed mode.

Definition at line 122 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.7 MMC_CLOCK_26MHZ

```
#define MMC_CLOCK_26MHZ
```

MMC card bus frequency 1 in high speed mode.

Definition at line 124 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.8 MMC_CLOCK_52MHZ

```
#define MMC_CLOCK_52MHZ
```

MMC card bus frequency 2 in high speed mode.

Definition at line 126 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.9 MMC_CLOCK_200MHZ

```
#define MMC_CLOCK_200MHZ
```

MMC card bus frequency 3 in HS200 or HS400 mode.

Definition at line 128 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.10 MMCSetBits

```
#define MMCSetBits
```

Access Set Bits field.

Bit fields to access EXT_CSD register

Definition at line 132 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.11 MMCClearBits

```
#define MMCClearBits
```

Access Clear Bits field.

Definition at line 134 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.12 MMCCWriteBits

```
#define MMCCWriteBits
```

Access Write Bits field.

Definition at line 136 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.13 MMCCCommandSets

```
#define MMCCCommandSets
```

Access Command Set field.

Definition at line 138 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.14 MMCSHiftIndex

```
#define MMCSHiftIndex
```

Shift index field.

Definition at line 140 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.15 MMCDefaultSpeed

```
#define MMCDefaultSpeed
```

Default Speed field.

Definition at line 142 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.16 MMCHightSpeed

```
#define MMCHightSpeed
```

High Speed field.

Definition at line 144 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.17 MMCHS200Speed

```
#define MMCHS200Speed
```

HS200 Speed field.

Definition at line 146 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.18 MMCHS400Speed

```
#define MMCHS400Speed
```

HS400 Speed field.

Definition at line 148 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.19 MMCPowerClass100

```
#define MMCPowerClass100
```

Power Class 100 field.

Definition at line 151 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.20 MMCPowerClass120

```
#define MMCPowerClass120
```

Power Class 120 field.

Definition at line 153 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.21 MMCPowerClass150

```
#define MMCPowerClass150
```

Power Class 150 field.

Definition at line 155 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.22 MMCPowerClass180

```
#define MMCPowerClass180
```

Power Class 180 field.

Definition at line 157 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.23 MMCPowerClass200

```
#define MMCPowerClass200
```

Power Class 200 field.

Definition at line 159 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.24 MMCPowerClass220

```
#define MMCPowerClass220
```

Power Class 220 field.

Definition at line 161 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.25 MMCPowerClass250

```
#define MMCPowerClass250
```

Power Class 250 field.

Definition at line 163 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.26 MMCPowerClass300

```
#define MMCPowerClass300
```

Power Class 300 field.

Definition at line 165 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.27 MMCPowerClass350

```
#define MMCPowerClass350
```

Power Class 350 field.

Definition at line 167 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.28 MMCPowerClass400

```
#define MMCPowerClass400
```

Power Class 400 field.

Definition at line 169 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.29 MMCPowerClass450

```
#define MMCPowerClass450
```

Power Class 450 field.

Definition at line 171 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.30 MMCPowerClass500

```
#define MMCPowerClass500
```

Power Class 500 field.

Definition at line 173 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.31 MMCPowerClass600

```
#define MMCPowerClass600
```

Power Class 600 field.

Definition at line 175 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.32 MMCPowerClass700

```
#define MMCPowerClass700
```

Power Class 700 field.

Definition at line 177 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.33 MMCPowerClass800

```
#define MMCPowerClass800
```

Power Class 800 field.

Definition at line 179 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.34 MMCPowerClass900

```
#define MMCPowerClass900
```

Power Class >800 field.

Definition at line 181 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.35 MMCBusWidth1

```
#define MMCBusWidth1
```

Bus width 1 bit field.

Definition at line 184 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.36 MMCBusWidth4

```
#define MMCBusWidth4
```

Bus width 4 bits field.

Definition at line 186 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.37 MMCBusWidth8

```
#define MMCBusWidth8
```

Bus width 8 bits field.

Definition at line 188 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.38 MMCBusWidth4ddr

```
#define MMCBusWidth4ddr
```

Bus width 4 bits ddr field.

Definition at line 190 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.39 MMCBusWidth8ddr

```
#define MMCBusWidth8ddr
```

Bus width 8 bit ddr field.

Definition at line 192 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.40 MMC_ALTERNATE_BOOT_SUPPORT_MASK

```
#define MMC_ALTERNATE_BOOT_SUPPORT_MASK
```

Alternate boot support(BOOT_INFO in Extended CSD)

Definition at line 195 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.41 MMC_POWER_CLASS_4BIT_MASK

```
#define MMC_POWER_CLASS_4BIT_MASK
```

The power class value bit mask when bus in 4 bit mode.

Definition at line 198 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.42 MMC_POWER_CLASS_8BIT_MASK

```
#define MMC_POWER_CLASS_8BIT_MASK
```

The power class current value bit mask when bus in 8 bit mode.

Definition at line 200 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.43 MMC_PARTITION_CONFIG_ACCESS_SHIFT

```
#define MMC_PARTITION_CONFIG_ACCESS_SHIFT
```

The bit shift for PARTITION ACCESS filed in PARTITION_CONFIG (PARTITION_CONFIG in Extend CSD)

Definition at line 203 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.44 MMC_PARTITION_CONFIG_ACCESS_MASK

```
#define MMC_PARTITION_CONFIG_ACCESS_MASK
```

The bit mask for PARTITION ACCESS field in PARTITION_CONFIG.

Definition at line 205 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.45 MMC_PARTITION_CONFIG_ACCESS

```
#define MMC_PARTITION_CONFIG_ACCESS(  
    x )
```

The bit set for PARTITION ACCESS field in PARTITION_CONFIG.

Definition at line 207 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.46 MMC_PARTITION_CONFIG_ENABLE_SHIFT

```
#define MMC_PARTITION_CONFIG_ENABLE_SHIFT
```

The bit shift for PARTITION ENABLE field in PARTITION_CONFIG.

Definition at line 210 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.47 MMC_PARTITION_CONFIG_ENABLE_MASK

```
#define MMC_PARTITION_CONFIG_ENABLE_MASK
```

The bit mask for PARTITION ENABLE field in PARTITION_CONFIG.

Definition at line 212 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.48 MMC_PARTITION_CONFIG_ENABLE

```
#define MMC_PARTITION_CONFIG_ENABLE(  
    x )
```

The bit set for PARTITION ENABLE field in PARTITION_CONFIG.

Definition at line 214 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.49 MMC_PARTITION_CONFIG_ACK_SHIFT

```
#define MMC_PARTITION_CONFIG_ACK_SHIFT
```

The bit shift for ACK field in PARTITION_CONFIG.

Definition at line 217 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.50 MMC_PARTITION_CONFIG_ACK_MASK

```
#define MMC_PARTITION_CONFIG_ACK_MASK
```

The bit mask for ACK field in PARTITION_CONFIG.

Definition at line 219 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.51 MMC_PARTITION_CONFIG_ACK

```
#define MMC_PARTITION_CONFIG_ACK(  
    x )
```

The bit set for ACK field in PARTITION_CONFIG.

Definition at line 221 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.52 MMC_BOOT_BUS_WIDTH_WIDTH_SHIFT

```
#define MMC_BOOT_BUS_WIDTH_WIDTH_SHIFT
```

The bit shift for BOOT BUS WIDTH field in BOOT_BUS_CONDITIONS.

Definition at line 224 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.53 MMC_BOOT_BUS_WIDTH_WIDTH_MASK

```
#define MMC_BOOT_BUS_WIDTH_WIDTH_MASK
```

The bit mask for BOOT BUS WIDTH field in BOOT_BUS_CONDITIONS.

Definition at line 226 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.54 MMC_BOOT_BUS_WIDTH_WIDTH

```
#define MMC_BOOT_BUS_WIDTH_WIDTH(  
    x )
```

The bit set for BOOT BUS WIDTH field in BOOT_BUS_CONDITIONS.

Definition at line 228 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.55 MMC_BOOT_BUS_WIDTH_RESET_SHIFT

```
#define MMC_BOOT_BUS_WIDTH_RESET_SHIFT
```

The bit shift for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.

Definition at line 231 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.56 MMC_BOOT_BUS_WIDTH_RESET_MASK

```
#define MMC_BOOT_BUS_WIDTH_RESET_MASK
```

The bit mask for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.

Definition at line 233 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.57 MMC_BOOT_BUS_WIDTH_RESET

```
#define MMC_BOOT_BUS_WIDTH_RESET(  
    x )
```

The bit set for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.

Definition at line 235 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.58 MMC_BOOT_MODE_SHIFT

```
#define MMC_BOOT_MODE_SHIFT
```

The bit shift for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.

Definition at line 238 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.59 MMC_BOOT_MODE_MASK

```
#define MMC_BOOT_MODE_MASK
```

The bit mask for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.

Definition at line 240 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.60 MMC_BOOT_MODE

```
#define MMC_BOOT_MODE(  
    x )
```

The bit set for BOOT BUS WIDTH RESET field in BOOT_BUS_CONDITIONS.

Definition at line 242 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.61 MMC_HS_BOOT_MODE_SHIFT

```
#define MMC_HS_BOOT_MODE_SHIFT
```

The bit shift for HS_BOOT_MODE field in BOOT_INFO.

Definition at line 245 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.62 MMC_HS_BOOT_MODE_MASK

```
#define MMC_HS_BOOT_MODE_MASK
```

The bit mask for HS_BOOT_MODE field in BOOT_INFO.

Definition at line 247 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.63 MMC_HS_BOOT_MODE

```
#define MMC_HS_BOOT_MODE(  
    x )
```

The bit set for HS_BOOT_MODE field in BOOT_INFO.

Definition at line 249 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.64 MMC_DDR_BOOT_MODE_SHIFT

```
#define MMC_DDR_BOOT_MODE_SHIFT
```

The bit shift for DDR_BOOT_MODE field in BOOT_INFO.

Definition at line 252 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.65 MMC_DDR_BOOT_MODE_MASK

```
#define MMC_DDR_BOOT_MODE_MASK
```

The bit mask for DDR_BOOT_MODE field in BOOT_INFO.

Definition at line 254 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.66 MMC_DDR_BOOT_MODE

```
#define MMC_DDR_BOOT_MODE(  
    x )
```

The bit set for DDR_BOOT_MODE field in BOOT_INFO.

Definition at line 256 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.67 SD_IP_R1_OUT_OF_RANGE_ERROR

```
#define SD_IP_R1_OUT_OF_RANGE_ERROR
```

Card status bit in R1.

(1 << 31) Out of range status bit

Definition at line 259 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.68 SD_IP_R1_ADDRESS_ERROR

```
#define SD_IP_R1_ADDRESS_ERROR
```

(1 << 30) Address error status bit

Definition at line 260 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.69 SD_IP_R1_BLOCK LENGHT_ERROR

```
#define SD_IP_R1_BLOCK LENGHT_ERROR
```

(1 << 29) Block length error status bit

Definition at line 261 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.70 SD_IP_R1_ERASE_SEQ_ERROR

```
#define SD_IP_R1_ERASE_SEQ_ERROR
```

(1 << 28) Erase sequence error status bit

Definition at line 262 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.71 SD_IP_R1_ERASE_PARAMETER_ERROR

```
#define SD_IP_R1_ERASE_PARAMETER_ERROR
```

(1 << 27) Erase parameter error status bit

Definition at line 263 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.72 SD_IP_R1_WRITE_PROTECT_VIOLATION

```
#define SD_IP_R1_WRITE_PROTECT_VIOLATION
```

(1 << 26) Write protection violation status bit

Definition at line 264 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.73 SD_IP_R1_CARD_IS_LOCKED

```
#define SD_IP_R1_CARD_IS_LOCKED
```

(1 << 25) Card locked status bit

Definition at line 265 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.74 SD_IP_R1_LOCK_UNLOCK_FAILED

```
#define SD_IP_R1_LOCK_UNLOCK_FAILED
```

(1 << 24) lock/unlock error status bit

Definition at line 266 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.75 SD_IP_R1_COMMAND_CRC_FAILED

```
#define SD_IP_R1_COMMAND_CRC_FAILED
```

(1 << 23) CRC error status bit

Definition at line 267 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.76 SD_IP_R1_ILLEGAL_COMMAND

```
#define SD_IP_R1_ILLEGAL_COMMAND
```

(1 << 22) Illegal command status bit

Definition at line 268 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.77 SD_IP_R1_CARD_ECC_FAILED

```
#define SD_IP_R1_CARD_ECC_FAILED
```

(1 << 21) Card ecc error status bit

Definition at line 269 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.78 SD_IP_R1_CARD_CONTROLLER_ERROR

```
#define SD_IP_R1_CARD_CONTROLLER_ERROR
```

(1 << 20) Internal card controller error status bit

Definition at line 270 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.79 SD_IP_R1_UNKNOWN_ERROR

```
#define SD_IP_R1_UNKNOWN_ERROR
```

(1 << 19) A general or an unknown error status bit

Definition at line 271 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.80 SD_IP_R1_CID_CSD_OVERWRITE

```
#define SD_IP_R1_CID_CSD_OVERWRITE
```

(1 << 16) Cid/csd overwrite status bit

Definition at line 272 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.81 SD_IP_R1_WRITE_PROTECT_ERASE_SKIP_FLAG

```
#define SD_IP_R1_WRITE_PROTECT_ERASE_SKIP_FLAG
```

(1 << 15) Write protection erase skip status bit

Definition at line 273 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.82 SD_IP_R1_CARD_ECC_DISABLED_FLAG

```
#define SD_IP_R1_CARD_ECC_DISABLED_FLAG
```

(1 << 14) Card ecc disabled status bit

Definition at line 274 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.83 SD_IP_R1_ERASE_RESET_FLAG

```
#define SD_IP_R1_ERASE_RESET_FLAG
```

(1 << 13) Erase reset status bit

Definition at line 275 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.84 SD_IP_R1_READY_FOR_DATA_FLAG

```
#define SD_IP_R1_READY_FOR_DATA_FLAG
```

(1 << 8) Ready for data status bit

Definition at line 276 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.85 SD_IP_R1_SWITCH_ERROR_FLAG

```
#define SD_IP_R1_SWITCH_ERROR_FLAG
```

(1 << 7) Switch error status bit

Definition at line 277 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.86 SD_IP_R1_APPLICATIONC_COMMAND_FLAG

```
#define SD_IP_R1_APPLICATIONC_COMMAND_FLAG
```

(1 << 5) Application command enabled status bit

Definition at line 278 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.87 SD_IP_R1_AUTHENTICATE_SEQ_ERROR

```
#define SD_IP_R1_AUTHENTICATE_SEQ_ERROR
```

(1 << 3) error in the sequence of authentication process

Definition at line 279 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.88 SD_IP_R1_CURRENT_STATE

```
#define SD_IP_R1_CURRENT_STATE(  
    x )
```

R1: current state.

Definition at line 286 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.89 aSD_OcrPowerUpBusyFlag

```
#define aSD_OcrPowerUpBusyFlag
```

OCR register in SD card.

(int)(1U << 31U)Power up busy status

Definition at line 390 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.90 aSD_OcrHostCapacitySupportFlag

```
#define aSD_OcrHostCapacitySupportFlag
```

(1U << 30U) Card capacity status

Definition at line 391 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.91 aSD_OcrCardCapacitySupportFlag

```
#define aSD_OcrCardCapacitySupportFlag
```

Card capacity status

Definition at line 392 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.92 aSD_OcrSwitch18RequestFlag

```
#define aSD_OcrSwitch18RequestFlag
```

(1U << 24U) Switch to 1.8V request

Definition at line 393 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.93 aSD_OcrSwitch18AcceptFlag

```
#define aSD_OcrSwitch18AcceptFlag
```

Switch to 1.8V accepted

Definition at line 394 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.94 aSD_OcrSwitchSDXCPowerControlFlag

```
#define aSD_OcrSwitchSDXCPowerControlFlag
```

(1U << 28U) SDXC Power Control

Definition at line 395 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.95 aSD_OcrVdd27_28Flag

```
#define aSD_OcrVdd27_28Flag
```

(1U << 15U) VDD 2.7-2.8

Definition at line 396 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.96 aSD_OcrVdd28_29Flag

```
#define aSD_OcrVdd28_29Flag
```

(1U << 16U) VDD 2.8-2.9

Definition at line 397 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.97 aSD_OcrVdd29_30Flag

```
#define aSD_OcrVdd29_30Flag
```

(1U << 17U) VDD 2.9-3.0

Definition at line 398 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.98 aSD_OcrVdd30_31Flag

```
#define aSD_OcrVdd30_31Flag
```

(1U << 18U) VDD 3.0-3.1

Definition at line 399 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.99 aSD_OcrVdd31_32Flag

```
#define aSD_OcrVdd31_32Flag
```

(1U << 19U) VDD 3.1-3.2

Definition at line 400 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.100 aSD_OcrVdd32_33Flag

```
#define aSD_OcrVdd32_33Flag
```

(1U << 20U) VDD 3.2-3.3

Definition at line 401 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.101 aSD_OcrVdd33_34Flag

```
#define aSD_OcrVdd33_34Flag
```

(1U << 21U) VDD 3.3-3.4

Definition at line 402 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.102 aSD_OcrVdd34_35Flag

```
#define aSD_OcrVdd34_35Flag
```

(1U << 22U) VDD 3.4-3.5

Definition at line 403 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.103 aSD_OcrVdd35_36Flag

```
#define aSD_OcrVdd35_36Flag
```

(1U << 23U) VDD 3.5-3.6

Definition at line 404 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.104 SD_IP_CSD_READ_BLOCK_PARTIAL_FLAG

```
#define SD_IP_CSD_READ_BLOCK_PARTIAL_FLAG
```

SD card CSD register flags.

(1U << 0U) Partial blocks for read allowed [79:79]

Definition at line 475 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.105 SD_IP_CSD_WRITE_BLOCK_MISALIGN_FLAG

```
#define SD_IP_CSD_WRITE_BLOCK_MISALIGN_FLAG
```

(1U << 1U) Write block misalignment [78:78]

Definition at line 476 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.106 SD_IP_CSD_READ_BLOCK_MISALIGN_FLAG

```
#define SD_IP_CSD_READ_BLOCK_MISALIGN_FLAG
```

(1U << 2U) Read block misalignment [77:77]

Definition at line 477 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.107 SD_IP_CSD_DSR_IMPLEMENTATION_FLAG

```
#define SD_IP_CSD_DSR_IMPLEMENTATION_FLAG
```

(1U << 3U) DSR implemented [76:76]

Definition at line 478 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.108 SD_IP_CSD_ERASE_BLOCK_ENABLED_FLAG

```
#define SD_IP_CSD_ERASE_BLOCK_ENABLED_FLAG
```

(1U << 4U) Erase single block enabled [46:46]

Definition at line 479 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.109 SD_IP_CSD_WRITE_PROTECT_GROUP_ENABLED_FLAG

```
#define SD_IP_CSD_WRITE_PROTECT_GROUP_ENABLED_FLAG
```

(1U << 5U) Write protect group enabled [31:31]

Definition at line 480 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.110 SD_IP_CSD_WRITE_BLOCK_PARTIAL_FLAG

```
#define SD_IP_CSD_WRITE_BLOCK_PARTIAL_FLAG
```

(1U << 6U) Partial blocks for write allowed [21:21]

Definition at line 481 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.111 SD_IP_CSD_FILE_FORMAT_GROUP_FLAG

```
#define SD_IP_CSD_FILE_FORMAT_GROUP_FLAG
```

(1U << 7U) File format group [15:15]

Definition at line 482 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.112 SD_IP_CSD_COPY_FLAG

```
#define SD_IP_CSD_COPY_FLAG
```

(1U << 8U) Copy flag [14:14]

Definition at line 483 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.113 SD_IP_CSD_PERMANENT_WRITE_PROTECT_FLAG

```
#define SD_IP_CSD_PERMANENT_WRITE_PROTECT_FLAG
```

(1U << 9U) Permanent write protection [13:13]

Definition at line 484 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.114 SD_IP_CSD_TEMPORARY_WRITE_PROTECT_FLAG

```
#define SD_IP_CSD_TEMPORARY_WRITE_PROTECT_FLAG
```

(1U << 10U) Temporary write protection [12:12]

Definition at line 485 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.115 MMC_OCR_V170TO195_SHIFT

```
#define MMC_OCR_V170TO195_SHIFT
```

The bit mask for VOLTAGE WINDOW 1.70V to 1.95V field in OCR.

Definition at line 488 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.116 MMC_OCR_V170TO195_MASK

```
#define MMC_OCR_V170TO195_MASK
```

The bit mask for VOLTAGE WINDOW 1.70V to 1.95V field in OCR.

Definition at line 490 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.117 MMC_OCR_V200TO260_SHIFT

```
#define MMC_OCR_V200TO260_SHIFT
```

The bit shift for VOLTAGE WINDOW 2.00V to 2.60V field in OCR.

Definition at line 492 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.118 MMC_OCR_V200TO260_MASK

```
#define MMC_OCR_V200TO260_MASK
```

The bit mask for VOLTAGE WINDOW 2.00V to 2.60V field in OCR.

Definition at line 494 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.119 MMC_OCR_V270TO360_SHIFT

```
#define MMC_OCR_V270TO360_SHIFT
```

The bit shift for VOLTAGE WINDOW 2.70V to 3.60V field in OCR.

Definition at line 496 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.120 MMC_OCR_V270TO360_MASK

```
#define MMC_OCR_V270TO360_MASK
```

The bit mask for VOLTAGE WINDOW 2.70V to 3.60V field in OCR.

Definition at line 498 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.121 MMC_OCR_ACCESS_MODE_SHIFT

```
#define MMC_OCR_ACCESS_MODE_SHIFT
```

The bit shift for ACCESS MODE field in OCR.

Definition at line 500 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.122 MMC_OCR_ACCESS_MODE_MASK

```
#define MMC_OCR_ACCESS_MODE_MASK
```

The bit mask for ACCESS MODE field in OCR.

Definition at line 502 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.123 MMC_OCR_SECTOR_ACCESS_MODE_MASK

```
#define MMC_OCR_SECTOR_ACCESS_MODE_MASK
```

The bit mask for SECTOR ACCESS MODE in OCR.

Definition at line 504 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.124 MMC_OCR_BUSY_SHIFT

```
#define MMC_OCR_BUSY_SHIFT
```

The bit shift for BUSY field in OCR.

Definition at line 506 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.125 MMC_OCR_BUSY_MASK

```
#define MMC_OCR_BUSY_MASK
```

The bit mask for BUSY field in OCR.

Definition at line 508 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.126 MMC_TRANSFER_FREQUENCY_UNIT_SHIFT

```
#define MMC_TRANSFER_FREQUENCY_UNIT_SHIFT
```

The bit shift for FREQUENCY UNIT field in TRANSFER SPEED(TRAN-SPEED in Extended CSD)

Definition at line 511 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.127 MMC_TRANSFER__FREQUENCY_UNIT_MASK

```
#define MMC_TRANSFER__FREQUENCY_UNIT_MASK
```

The bit mask for FREQUENCY UNIT in TRANSFER SPEED.

Definition at line 513 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.128 MMC_TRANSFER_MULTIPLIER_SHIFT

```
#define MMC_TRANSFER_MULTIPLIER_SHIFT
```

The bit shift for MULTIPLIER field in TRANSFER SPEED.

Definition at line 515 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.129 MMC_TRANSFER_MULTIPLIER_MASK

```
#define MMC_TRANSFER_MULTIPLIER_MASK
```

The bit mask for MULTIPLIER field in TRANSFER SPEED

Definition at line 517 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.130 READ_MMC_TRANSFER_SPEED_FREQUENY_UNIT

```
#define READ_MMC_TRANSFER_SPEED_FREQUENY_UNIT(  
    CSD )
```

Read the value of FREQUENCY UNIT in TRANSFER SPEED.

Definition at line 520 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.131 READ_MMC_TRANSFER_SPEED_MULTIPLIER

```
#define READ_MMC_TRANSFER_SPEED_MULTIPLIER(  
    CSD )
```

Read the value of MULTIPLIER field in TRANSFER SPEED.

Definition at line 523 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.132 MMC_EXT_CSD_SEC_CNT_SHIFT

```
#define MMC_EXT_CSD_SEC_CNT_SHIFT
```

The bit shift for MMC_EXT_CSD_SEC_CNT field in extended CSD register.

Definition at line 527 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.133 MMC_EXT_CSD_DATA_SECTOR_SIZE_SHIFT

```
#define MMC_EXT_CSD_DATA_SECTOR_SIZE_SHIFT
```

The bit shift for MMC_EXT_CSD_DATA_SECTOR_SIZE field in extended CSD register.

Definition at line 529 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.134 MMC_EXT_CSD_BUS_WIDTH_SHIFT

```
#define MMC_EXT_CSD_BUS_WIDTH_SHIFT
```

The bit shift for MMC_EXT_CSD_BUS_WIDTH field in extended CSD register.

Definition at line 531 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.135 MMC_EXT_CSD_REV_SHIFT

```
#define MMC_EXT_CSD_REV_SHIFT
```

The bit shift for MMC_EXT_CSD_REV field in extended CSD register.

Definition at line 533 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.136 MMC_EXT_CSD_STRUCTURE_SHIFT

```
#define MMC_EXT_CSD_STRUCTURE_SHIFT
```

The bit shift for MMC_EXT_CSD_STRUCTURE field in extended CSD register.

Definition at line 535 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.137 MMC_EXT_CSD_CARD_TYPE_SHIFT

```
#define MMC_EXT_CSD_CARD_TYPE_SHIFT
```

The bit shift for MMC_EXT_CSD_CARD_TYPE field in extended CSD register.

Definition at line 537 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.138 MMC_EXT_CSD_CARD_TYPE_MASK

```
#define MMC_EXT_CSD_CARD_TYPE_MASK
```

The bit mask for MMC_EXT_CSD_CARD_TYPE field in extended CSD register.

Definition at line 539 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.139 MMC_EXT_CSD_STROBE_SUPPORT_SHIFT

```
#define MMC_EXT_CSD_STROBE_SUPPORT_SHIFT
```

The bit shift for MMC_EXT_CSD_STROBE_SUPPORT field in extended CSD register.

Definition at line 541 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.140 MMC_EXT_CSD_HS_TIMING_SHIFT

```
#define MMC_EXT_CSD_HS_TIMING_SHIFT
```

The bit shift for MMC_EXT_CSD_STROBE_SUPPORT field in extended CSD register.

Definition at line 543 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.141 MMC_EXT_CSD_PARTITION_CONFIG_SHIFT

```
#define MMC_EXT_CSD_PARTITION_CONFIG_SHIFT
```

The bit shift for MMC_EXT_CSD_INDEX_PARTITION_CONFIG field in extended CSD register.

Definition at line 546 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.142 MMC_EXT_CSD_BOOT_SIZE_MULT_SHIFT

```
#define MMC_EXT_CSD_BOOT_SIZE_MULT_SHIFT
```

The bit shift for MMC_EXT_CSD_INDEX_BOOT_SIZE_MULT field in extended CSD register.

Definition at line 548 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.143 MMC_EXT_CSD_BOOT_INFO_SHIFT

```
#define MMC_EXT_CSD_BOOT_INFO_SHIFT
```

The bit shift for MMC_EXT_CSD_INDEX_BOOT_INFO field in extended CSD register.

Definition at line 550 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.144 MMC_EXT_CSD_BOOT_BUS_CONDITIONS_SHIFT

```
#define MMC_EXT_CSD_BOOT_BUS_CONDITIONS_SHIFT
```

The bit shift for MMC_EXT_CSD_INDEX_BOOT_BUS_CONDITIONS field in extended CSD register.

Definition at line 552 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.145 MMC_EXT_CSD_INDEX_FLUSH_CACHE

```
#define MMC_EXT_CSD_INDEX_FLUSH_CACHE
```

EXT CSD byte index.

W

Definition at line 571 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.146 MMC_EXT_CSD_INDEX_CACHE_CTRL

```
#define MMC_EXT_CSD_INDEX_CACHE_CTRL
```

R/W

Definition at line 572 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.147 MMC_EXT_CSD_INDEX_POWER_OFF_NOTIFICATION

```
#define MMC_EXT_CSD_INDEX_POWER_OFF_NOTIFICATION
```

R/W

Definition at line 573 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.148 MMC_EXT_CSD_INDEX_EXP_EVENTS_STATUS

```
#define MMC_EXT_CSD_INDEX_EXP_EVENTS_STATUS
```

RO, 2 bytes

Definition at line 574 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.149 MMC_EXT_CSD_INDEX_DATA_SECTOR_SIZE

```
#define MMC_EXT_CSD_INDEX_DATA_SECTOR_SIZE
```

R

Definition at line 575 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.150 MMC_EXT_CSD_INDEX_NATIVE_SECTOR_SIZE

```
#define MMC_EXT_CSD_INDEX_NATIVE_SECTOR_SIZE
```

R

Definition at line 576 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.151 MMC_EXT_CSD_INDEX_GP_SIZE_MULT

```
#define MMC_EXT_CSD_INDEX_GP_SIZE_MULT
```

R/W, 12 bytes

Definition at line 577 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.152 MMC_EXT_CSD_INDEX_PARTITION_ATTRIBUTE

```
#define MMC_EXT_CSD_INDEX_PARTITION_ATTRIBUTE
```

R/W

Definition at line 578 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.153 MMC_EXT_CSD_INDEX_PARTITION_SUPPORT

```
#define MMC_EXT_CSD_INDEX_PARTITION_SUPPORT
```

RO

Definition at line 579 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.154 MMC_EXT_CSD_INDEX_HPI_MGMT

```
#define MMC_EXT_CSD_INDEX_HPI_MGMT
```

R/W

Definition at line 580 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.155 MMC_EXT_CSD_INDEX_RST_N_FUNCTION

```
#define MMC_EXT_CSD_INDEX_RST_N_FUNCTION
```

R/W

Definition at line 581 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.156 MMC_EXT_CSD_INDEX_BKOPS_EN

```
#define MMC_EXT_CSD_INDEX_BKOPS_EN
```

R/W

Definition at line 582 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.157 MMC_EXT_CSD_INDEX_BKOPS_START

```
#define MMC_EXT_CSD_INDEX_BKOPS_START
```

W

Definition at line 583 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.158 MMC_EXT_CSD_INDEX_SANITIZE_START

```
#define MMC_EXT_CSD_INDEX_SANITIZE_START
```

W

Definition at line 584 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.159 MMC_EXT_CSD_INDEX_WR_REL_PARAM

```
#define MMC_EXT_CSD_INDEX_WR_REL_PARAM
```

RO

Definition at line 585 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.160 MMC_EXT_CSD_INDEX_USER_WP

```
#define MMC_EXT_CSD_INDEX_USER_WP
```

R/W

Definition at line 586 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.161 MMC_EXT_CSD_INDEX_BOOT_WP

```
#define MMC_EXT_CSD_INDEX_BOOT_WP
```

R/W

Definition at line 587 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.162 MMC_EXT_CSD_INDEX_BOOT_WP_STATUS

```
#define MMC_EXT_CSD_INDEX_BOOT_WP_STATUS
```

R/W

Definition at line 588 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.163 MMC_EXT_CSD_INDEX_ERASE_GROUP_DEF

```
#define MMC_EXT_CSD_INDEX_ERASE_GROUP_DEF
```

R/W

Definition at line 589 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.164 MMC_EXT_CSD_INDEX_BOOT_BUS_CONDITIONS

```
#define MMC_EXT_CSD_INDEX_BOOT_BUS_CONDITIONS
```

R/W

Definition at line 590 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.165 MMC_EXT_CSD_INDEX_BOOT_CONFIG_PROT

```
#define MMC_EXT_CSD_INDEX_BOOT_CONFIG_PROT
```

R/W

Definition at line 591 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.166 MMC_EXT_CSD_INDEX_PARTITION_CONFIG

```
#define MMC_EXT_CSD_INDEX_PARTITION_CONFIG
```

R/W

Definition at line 592 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.167 MMC_EXT_CSD_INDEX_ERASED_MEM_CONT

```
#define MMC_EXT_CSD_INDEX_ERASED_MEM_CONT
```

RO

Definition at line 593 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.168 MMC_EXT_CSD_INDEX_BUS_WIDTH

```
#define MMC_EXT_CSD_INDEX_BUS_WIDTH
```

R/W

Definition at line 594 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.169 MMC_EXT_CSD_STROBE_SUPPORT

```
#define MMC_EXT_CSD_STROBE_SUPPORT
```

R/W

Definition at line 595 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.170 MMC_EXT_CSD_INDEX_HS_TIMING

```
#define MMC_EXT_CSD_INDEX_HS_TIMING
```

R/W

Definition at line 596 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.171 MMC_EXT_CSD_INDEX_POWER_CLASS

```
#define MMC_EXT_CSD_INDEX_POWER_CLASS
```

R/W

Definition at line 597 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.172 MMC_EXT_CSD_INDEX_REV

```
#define MMC_EXT_CSD_INDEX_REV
```

RO

Definition at line 598 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.173 MMC_EXT_CSD_INDEX_STRUCTURE

```
#define MMC_EXT_CSD_INDEX_STRUCTURE
```

RO

Definition at line 599 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.174 MMC_EXT_CSD_INDEX_CARD_TYPE

```
#define MMC_EXT_CSD_INDEX_CARD_TYPE
```

RO

Definition at line 600 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.175 MMC_EXT_CSD_INDEX_OUT_OF_INTERRUPT_TIME

```
#define MMC_EXT_CSD_INDEX_OUT_OF_INTERRUPT_TIME
```

RO

Definition at line 601 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.176 MMC_EXT_CSD_INDEX_PART_SWITCH_TIME

```
#define MMC_EXT_CSD_INDEX_PART_SWITCH_TIME
```

RO

Definition at line 602 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.177 MMC_EXT_CSD_INDEX_PWR_CL_52_195

```
#define MMC_EXT_CSD_INDEX_PWR_CL_52_195
```

RO

Definition at line 603 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.178 MMC_EXT_CSD_INDEX_PWR_CL_26_195

```
#define MMC_EXT_CSD_INDEX_PWR_CL_26_195
```

RO

Definition at line 604 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.179 MMC_EXT_CSD_INDEX_PWR_CL_52_360

```
#define MMC_EXT_CSD_INDEX_PWR_CL_52_360
```

RO

Definition at line 605 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.180 MMC_EXT_CSD_INDEX_PWR_CL_26_360

```
#define MMC_EXT_CSD_INDEX_PWR_CL_26_360
```

RO

Definition at line 606 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.181 MMC_EXT_CSD_INDEX_SEC_CNT

```
#define MMC_EXT_CSD_INDEX_SEC_CNT
```

RO, 4 bytes

Definition at line 607 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.182 MMC_EXT_CSD_INDEX_S_A_TIMEOUT

```
#define MMC_EXT_CSD_INDEX_S_A_TIMEOUT
```

RO

Definition at line 608 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.183 MMC_EXT_CSD_INDEX_REL_WR_SEC_C

```
#define MMC_EXT_CSD_INDEX_REL_WR_SEC_C
```

RO

Definition at line 609 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.184 MMC_EXT_CSD_INDEX_HC_WP_GRP_SIZE

```
#define MMC_EXT_CSD_INDEX_HC_WP_GRP_SIZE
```

RO

Definition at line 610 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.185 MMC_EXT_CSD_INDEX_ERASE_TIMEOUT_MULT

```
#define MMC_EXT_CSD_INDEX_ERASE_TIMEOUT_MULT
```

RO

Definition at line 611 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.186 MMC_EXT_CSD_INDEX_HC_ERASE_GRP_SIZE

```
#define MMC_EXT_CSD_INDEX_HC_ERASE_GRP_SIZE
```

RO

Definition at line 612 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.187 MMC_EXT_CSD_INDEX_BOOT_SIZE_MULT

```
#define MMC_EXT_CSD_INDEX_BOOT_SIZE_MULT
```

RO

Definition at line 613 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.188 MMC_EXT_CSD_INDEX_BOOT_INFO

```
#define MMC_EXT_CSD_INDEX_BOOT_INFO
```

RO

Definition at line 614 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.189 MMC_EXT_CSD_INDEX_SEC_TRIM_MULT

```
#define MMC_EXT_CSD_INDEX_SEC_TRIM_MULT
```

RO

Definition at line 615 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.190 MMC_EXT_CSD_INDEX_SEC_ERASE_MULT

```
#define MMC_EXT_CSD_INDEX_SEC_ERASE_MULT
```

RO

Definition at line 616 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.191 MMC_EXT_CSD_INDEX_SEC_FEATURE_SUPPORT

```
#define MMC_EXT_CSD_INDEX_SEC_FEATURE_SUPPORT
```

RO

Definition at line 617 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.192 MMC_EXT_CSD_INDEX_TRIM_MULT

```
#define MMC_EXT_CSD_INDEX_TRIM_MULT
```

RO

Definition at line 618 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.193 MMC_EXT_CSD_INDEX_PWR_CL_200_130

```
#define MMC_EXT_CSD_INDEX_PWR_CL_200_130
```

RO

Definition at line 619 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.194 MMC_EXT_CSD_INDEX_PWR_CL_200_195

```
#define MMC_EXT_CSD_INDEX_PWR_CL_200_195
```

RO

Definition at line 620 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.195 MMC_EXT_CSD_INDEX_PWR_CL_DDR_52_195

```
#define MMC_EXT_CSD_INDEX_PWR_CL_DDR_52_195
```

RO

Definition at line 621 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.196 MMC_EXT_CSD_INDEX_PWR_CL_DDR_52_360

```
#define MMC_EXT_CSD_INDEX_PWR_CL_DDR_52_360
```

RO

Definition at line 622 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.197 MMC_EXT_CSD_INDEX_BKOPS_STATUS

```
#define MMC_EXT_CSD_INDEX_BKOPS_STATUS
```

RO

Definition at line 623 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.198 MMC_EXT_CSD_INDEX_POWER_OFF_LONG_TIME

```
#define MMC_EXT_CSD_INDEX_POWER_OFF_LONG_TIME
```

RO

Definition at line 624 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.199 MMC_EXT_CSD_INDEX_GENERIC_CMD6_TIME

```
#define MMC_EXT_CSD_INDEX_GENERIC_CMD6_TIME
```

RO

Definition at line 625 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.200 MMC_EXT_CSD_INDEX_CACHE_SIZE

```
#define MMC_EXT_CSD_INDEX_CACHE_SIZE
```

RO, 4 bytes

Definition at line 626 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.201 MMC_EXT_CSD_INDEX_TAG_UNIT_SIZE

```
#define MMC_EXT_CSD_INDEX_TAG_UNIT_SIZE
```

RO

Definition at line 627 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.202 MMC_EXT_CSD_INDEX_DATA_TAG_SUPPORT

```
#define MMC_EXT_CSD_INDEX_DATA_TAG_SUPPORT
```

RO

Definition at line 628 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.203 MMC_EXT_CSD_INDEX_MAX_PACKED_WRITES

```
#define MMC_EXT_CSD_INDEX_MAX_PACKED_WRITES
```

RO

Definition at line 629 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.204 MMC_EXT_CSD_INDEX_MAX_PACKED_READS

```
#define MMC_EXT_CSD_INDEX_MAX_PACKED_READS
```

RO

Definition at line 630 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.205 MMC_EXT_CSD_INDEX_BKOPS_SUPPORT

```
#define MMC_EXT_CSD_INDEX_BKOPS_SUPPORT
```

RO

Definition at line 631 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.206 MMC_EXT_CSD_INDEX_HPI_FEATURES

```
#define MMC_EXT_CSD_INDEX_HPI_FEATURES
```

RO

Definition at line 632 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.207 MMC_EXTENDED_CSD_BYTES

```
#define MMC_EXTENDED_CSD_BYTES
```

The length of Extended CSD register, unit as bytes.

Definition at line 635 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.208 MMC_DEFAULT_RELATIVE_ADDRESS

```
#define MMC_DEFAULT_RELATIVE_ADDRESS
```

MMC card default relative address.

Definition at line 638 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.209 SD_PRODUCT_NAME_BYTES

```
#define SD_PRODUCT_NAME_BYTES
```

SD card product name length united as bytes.

Definition at line 641 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.210 MMC_PRODUCT_NAME_BYTES

```
#define MMC_PRODUCT_NAME_BYTES
```

MMC card product name length united as bytes.

Definition at line 644 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.211 SD_TRANSFER_SPEED_RATE_UNIT_SHIFT

```
#define SD_TRANSFER_SPEED_RATE_UNIT_SHIFT
```

The bit shift for RATE UNIT field in TRANSFER SPEED.

Definition at line 694 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.212 SD_TRANSFER_SPEED_RATE_UNIT_MASK

```
#define SD_TRANSFER_SPEED_RATE_UNIT_MASK
```

The bit mask for RATE UNIT field in TRANSFER SPEED.

Definition at line 696 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.213 SD_TRANSFER_SPEED_TIME_VALUE_SHIFT

```
#define SD_TRANSFER_SPEED_TIME_VALUE_SHIFT
```

The bit shift for TIME VALUE field in TRANSFER SPEED.

Definition at line 698 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.214 SD_TRANSFER_SPEED_TIME_VALUE_MASK

```
#define SD_TRANSFER_SPEED_TIME_VALUE_MASK
```

The bit mask for TIME VALUE field in TRANSFER SPEED.

Definition at line 700 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.215 SD_RD_TRANSFER_SPEED_RATE_UNIT

```
#define SD_RD_TRANSFER_SPEED_RATE_UNIT(  
    x )
```

Read the value of FREQUENCY UNIT in TRANSFER SPEED field.

Definition at line 702 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.216 SD_RD_TRANSFER_SPEED_TIME_VALUE

```
#define SD_RD_TRANSFER_SPEED_TIME_VALUE(  
    x )
```

Read the value of TIME VALUE in TRANSFER SPEED field.

Definition at line 705 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.217 MMC_SWITCH_COMMAND_SET_SHIFT

```
#define MMC_SWITCH_COMMAND_SET_SHIFT
```

The bit shift for COMMAND SET field in SWITCH command.

Definition at line 792 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.218 MMC_SWITCH_COMMAND_SET_MASK

```
#define MMC_SWITCH_COMMAND_SET_MASK
```

The bit mask for COMMAND set field in SWITCH command.

Definition at line 794 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.219 MMC_SWITCH_VALUE_SHIFT

```
#define MMC_SWITCH_VALUE_SHIFT
```

The bit shift for VALUE field in SWITCH command.

Definition at line 796 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.220 MMC_SWITCH_VALUE_MASK

```
#define MMC_SWITCH_VALUE_MASK
```

The bit mask for VALUE field in SWITCH command.

Definition at line 798 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.221 MMC_SWITCH_BYTE_INDEX_SHIFT

```
#define MMC_SWITCH_BYTE_INDEX_SHIFT
```

The bit shift for BYTE INDEX field in SWITCH command.

Definition at line 800 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.222 MMC_SWITCH_BYTE_INDEX_MASK

```
#define MMC_SWITCH_BYTE_INDEX_MASK
```

The bit mask for BYTE INDEX field in SWITCH command.

Definition at line 802 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.223 MMC_SWITCH_ACCESS_MODE_SHIFT

```
#define MMC_SWITCH_ACCESS_MODE_SHIFT
```

The bit shift for ACCESS MODE field in SWITCH command.

Definition at line 804 of file [Sd_Emmc_Ip_Types.h](#).

6.2.3.224 MMC_SWITCH_ACCESS_MODE_MASK

```
#define MMC_SWITCH_ACCESS_MODE_MASK
```

The bit mask for ACCESS MODE field in SWITCH command.

Definition at line 806 of file [Sd_Emmc_Ip_Types.h](#).

6.2.4 Types Reference**6.2.4.1 Sd_Ip_ProtocolType**

```
typedef uint8 Sd_Ip_ProtocolType
```

SD or MMC card type: memory card type that can be plugged into the SD port.

Definition at line 843 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5 Enum Reference**6.2.5.1 Sd_Emmc_Ip_StatusType**

```
enum Sd_Emmc_Ip_StatusType
```

SD/MMC generic status.

Enumerator

SD_IP_STATUS_SUCCESS	Generic operation success status
SD_IP_STATUS_PENDING	Generic operation success status
SD_IP_STATUS_ERROR	Generic operation failure status
SD_IP_STATUS_TIMEOUT	Generic operation timeout status
SD_IP_STATUS_NOT_SUPPORTED	Ip is noto supported
SD_IP_STATUS_TRANSFER_FAILED	Send command failed
SD_IP_STATUS_SET_CARD_BLOCK_SIZE_FAILED	Set block size failed
SD_IP_STATUS_CARD_NOT_SUPPORTED	Card doesn't support
SD_IP_STATUS_ALL_SEND_CID_FAILED	Send CID failed
SD_IP_STATUS_SEND_RELATIVE_ADDRESS_FAILED	Send relative address failed
SD_IP_STATUS_SEND_CSD_FAILED	Send CSD failed
SD_IP_STATUS_SELECT_CARD_FAILED	Select card failed
SD_IP_STATUS_SEND_SCR_FAILED	Send SCR failed
SD_IP_STATUS_SET_DATA_BUS_WIDTH_FAILED	Set bus width failed
SD_IP_STATUS_GO_IDLE_FAILED	Go idle failed
SD_IP_STATUS_HANDSHAKE_OP_FAILED	Send Operation Condition failed
SD_IP_STATUS_SEND_APPL_COMMAND_FAILED	Send application command failed
SD_IP_STATUS_SWITCH_FAILED	Switch command failed
SD_IP_STATUS_STOP_TRANSMISION_FAILED	Stop transmission failed
SD_IP_STATUS_WAIT_WRITE_COMPLETE_FAILED	Wait write complete failed
SD_IP_STATUS_SET_BLOCK_COUNT_FAILED	Set block count failed
SD_IP_STATUS_SWITCH_HIGH_SPEED_FAILED	Switch high speed failed
SD_IP_STATUS_CARD_IS_WRITE_PROTECTED	Card write protection lock enabled
SD_IP_STATUS_SWITCH_VOLTAGE_FAILED	Card switch voltage failed
SD_IP_STATUS_BLOCK_SIZE_INCORRECT	Block size is not as defined in HLD
SD_IP_STATUS_BLOCK_COUNT_INCORRECT	Block count is not as defined in HLD
SD_IP_STATUS_MEMORY_CARD_INCORRECT	Memory card type is not as defined in HLD
SD_IP_STATUS_TUNING_FAILED	Tuning data does not match with pattern

Definition at line 78 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.2 Sd_Ip_R1CurrentStateType

enum [Sd_Ip_R1CurrentStateType](#)

CURRENT_STATE filed in R1.

Enumerator

aSDMMC_R1StateIdle	R1: current state: idle
aSDMMC_R1StateReady	R1: current state: ready

Enumerator

aSDMMC_R1StateIdentify	R1: current state: identification
aSDMMC_R1StateStandby	R1: current state: standby
aSDMMC_R1StateTransfer	R1: current state: transfer
aSDMMC_R1StateSendData	R1: current state: sending data
aSDMMC_R1StateReceiveData	R1: current state: receiving data
aSDMMC_R1StateProgram	R1: current state: programming
aSDMMC_R1StateDisconnect	R1: current state: disconnect

Definition at line 289 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.3 Sd_Ip_CommandType

enum [Sd_Ip_CommandType](#)

SD card individual commands.

Enumerator

aSD_SendRelativeAddress	Send Relative Address
aSD_Switch	Switch Function
aSD_SendInterfaceCondition	Send Interface Condition
aSD_VoltageSwitch	Voltage Switch
aSD_SpeedClassControl	Speed Class control
aSD_EraseWriteBlockStart	Write Block Start
aSD_EraseWriteBlockEnd	Write Block End

Definition at line 303 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.4 Sd_Ip_ApplicationCmdType

enum [Sd_Ip_ApplicationCmdType](#)

SD card individual application commands.

Enumerator

aSD_ApplicationSetBusWidth	Set Bus Width
aSD_ApplicationStatus	Send SD status
aSD_ApplicationSendNumberWriteBlocks	Send Number Of Written Blocks

Enumerator

aSD_ApplicationSetWriteBlockEraseCount	Set Write Block Erase Count
aSD_ApplicationSendOperationCondition	Send Operation Condition
aSD_ApplicationSetClearCardDetect	Set Connect/Disconnect pull up on detect pin
aSD_ApplicationSendScr	Send Scr

Definition at line 315 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.5 Sd_Ip_SDMMC_CommandType

enum [Sd_Ip_SDMMC_CommandType](#)

SD/MMC card common commands.

Enumerator

aSDMMC_GoIdleState	Go Idle State
aSDMMC_AllSendCid	All Send CID
aSDMMC_SetDsr	Set DSR
aSDMMC_SelectCard	Select Card
aSDMMC_SendCsd	Send CSD
aSDMMC_SendCid	Send CID
aSDMMC_StopTransmission	Stop Transmission
aSDMMC_SendStatus	Send Status
aSDMMC_GoInactiveState	Go Inactive State
aSDMMC_SetBlockLength	Set Block Length
aSDMMC_ReadSingleBlock	Read Single Block
aSDMMC_ReadMultipleBlock	Read Multiple Block
aSDMMC_SendTuningBlock	Send Tuning Block
aSDMMC_SendTuningPattern	Send Tuning Pattern HS200
aSDMMC_SetBlockCount	Set Block Count
aSDMMC_WriteSingleBlock	Write Single Block
aSDMMC_WriteMultipleBlock	Write Multiple Block
aSDMMC_ProgramCsd	Program CSD
aSDMMC_SetWriteProtect	Set Write Protect
aSDMMC_ClearWriteProtect	Clear Write Protect
aSDMMC_SendWriteProtect	Send Write Protect
aSDMMC_Erase	Erase
aSDMMC_LockUnlock	Lock Unlock
aSDMMC_ApplicationCommand	Send Application Command
aSDMMC_GeneralCommand	General Purpose Command
aSDMMC_ReadOcr	Read OCR

Definition at line 327 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.6 Sd_Ip_MmcCommandType

enum [Sd_Ip_MmcCommandType](#)

MMC card individual commands.

Enumerator

aMMC_SendOperationCondition	Send Operation Condition
aMMC_SetRelativeAddress	Set Relative Address
aMMC_SleepAwake	Sleep Awake
aMMC_Switch	Switch
aMMC_SendExtendedCsd	Send EXT_CSD
aMMC_ReadDataUntilStop	Read Data Until Stop
aMMC_BusTestRead	Test Read
aMMC_WriteDataUntilStop	Write Data Until Stop
aMMC_ProgramCid	Program CID
aMMC_EraseGroupStart	Erase Group Start
aMMC_EraseGroupEnd	Erase Group End
aMMC_FastInputOutput	Fast IO
aMMC_GoInterruptState	Go interrupt State

Definition at line 358 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.7 Sd_Ip_CommandClassType

enum [Sd_Ip_CommandClassType](#)

SD card command class.

Enumerator

aSDMMC_CommandClassBasic	(1U << 0U) Card command class 0
aSDMMC_CommandClassBlockRead	(1U << 2U) Card command class 2
aSDMMC_CommandClassBlockWrite	(1U << 4U) Card command class 4
aSDMMC_CommandClassErase	(1U << 5U) Card command class 5
aSDMMC_CommandClassWriteProtect	(1U << 6U) Card command class 6
aSDMMC_CommandClassLockCard	(1U << 7U) Card command class 7
aSDMMC_CommandClassApplicationSpecific	(1U << 8U) Card command class 8
aSDMMC_CommandClassInputOutputMode	(1U << 9U) Card command class 9
aSDMMC_CommandClassSwitch	(1U << 10U) Card command class 10

Definition at line 376 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.8 Sd_Ip_SpecificationVersionType

enum [Sd_Ip_SpecificationVersionType](#)

SD card specification version number.

Enumerator

aSD_SpecificationVersion1↔ _0	(1U << 0U) SD card version 1.0-1.01
aSD_SpecificationVersion1↔ _1	(1U << 1U) SD card version 1.10
aSD_SpecificationVersion2↔ _0	(1U << 2U) SD card version 2.00
aSD_SpecificationVersion3↔ _0	(1U << 3U) SD card version 3.0

Definition at line 407 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.9 Sd_Ip_DataBusWidthType

enum [Sd_Ip_DataBusWidthType](#)

SD card bus width.

Enumerator

aSD_DataBusWidth1Bit	(1U << 0U) SD data bus width 1-bit mode
aSD_DataBusWidth4Bit	(1U << 2U) SD data bus width 4-bit mode

Definition at line 416 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.10 Sd_Ip_SwitchModeType

enum [Sd_Ip_SwitchModeType](#)

SD card switch mode.

Enumerator

aSD_SwitchCheck	SD switch mode 0: check function
aSD_SwitchSet	SD switch mode 1: set function

Definition at line 423 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.11 Sd_Ip_TimingFunctionType

enum [Sd_Ip_TimingFunctionType](#)

SD timing function number.

Enumerator

aSD_FunctionSDR12Default	SDR12 mode & default
aSD_FunctionSDR25HighSpeed	SDR25 & high speed
aSD_FunctionSDR50	SDR50 mode
aSD_FunctionSDR104	SDR104 mode
aSD_FunctionDDR50	DDR50 mode

Definition at line 430 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.12 Sd_Ip_GroupNumType

enum [Sd_Ip_GroupNumType](#)

SD group number.

Enumerator

aSD_GroupTimingMode	access mode group
aSD_GroupCommandSystem	command system group
aSD_GroupDriverStrength	driver strength group
aSD_GroupCurrentLimit	current limit group

Definition at line 440 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.13 Sd_Ip_DriverStrenghtType

enum [Sd_Ip_DriverStrenghtType](#)

SD card driver strength.

Enumerator

aSD_DriverStrengthTypeB	default driver strength
aSD_DriverStrengthTypeA	driver strength TYPE A
aSD_DriverStrengthTypeC	driver strength TYPE C
aSD_DriverStrengthTypeD	driver strength TYPE D

Definition at line 449 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.14 Sd_Ip_MaxCurrentType

enum [Sd_Ip_MaxCurrentType](#)

SD card current limit.

Enumerator

aSD_CurrentLimit200MA	default current limit
aSD_CurrentLimit400MA	current limit to 400MA
aSD_CurrentLimit600MA	current limit to 600MA
aSD_CurrentLimit800MA	current limit to 800MA

Definition at line 458 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.15 Sd_Ip_ScrFlagType

enum [Sd_Ip_ScrFlagType](#)

SD card SCR register flags.

Enumerator

aSD_ScrDataStatusAfterErase	(1U << 0U) Data status after erases [55:55]
aSD_ScrSdSpecification3	(1U << 1U) Specification version 3.00 or higher [47:47]

Definition at line 467 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.16 Sd_Ip_MmcECsdFlagType

enum [Sd_Ip_MmcECsdFlagType](#)

MMC card CSD register flags.

Enumerator

aMMC_CsdReadBlockPartialFlag	(1U << 0U) Partial blocks for read allowed
aMMC_CsdWriteBlockMisalignFlag	(1U << 1U) Write block misalignment
aMMC_CsdReadBlockMisalignFlag	(1U << 2U) Read block misalignment
aMMC_CsdDsrImplementedFlag	(1U << 3U) DSR implemented
aMMC_CsdWriteProtectGroupEnabledFlag	(1U << 4U) Write protect group enabled
aMMC_CsdWriteBlockPartialFlag	(1U << 5U) Partial blocks for write allowed
aMMC_ContentProtectApplicationFlag	(1U << 6U) Content protect application
aMMC_CsdFileFormatGroupFlag	(1U << 7U) File format group
aMMC_CsdCopyFlag	(1U << 8U) Copy flag
aMMC_CsdPermanentWriteProtectFlag	(1U << 9U) Permanent write protection
aMMC_CsdTemporaryWriteProtectFlag	(1U << 10U) Temporary write protection

Definition at line 555 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.17 Sd_Ip_CardFlagType

enum [Sd_Ip_CardFlagType](#)

Memory card flags.

Enumerator

SD_IP_SUPPORT_SPEED_CLASS_CONTROL_CMD	Card supports speed class
SD_IP_SUPPORT_SET_BLOCK_COUNT_CMD	Card supports CMD23
SD_IP_SUPPORT_4_BIT_WIDTH_FLAG	Support 4-bit data width
SD_IP_SUPPORT_HIGH_CAPACITY_FLAG	Support high capacity
SD_IP_SUPPORT_SDHC_FLAG	Card is SDHC
SD_IP_SUPPORT_SDXC_FLAG	Card is SDXC
SD_IP_WRITE_PROTECT_FLAG	Card has write protection lock enabled
SD_IP_SUPPORT_VOLTAGE_180V	Card supports 1.8V

Definition at line 822 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.18 Sd_Ip_MmcPartitionAccessType

enum [Sd_Ip_MmcPartitionAccessType](#)

MMC card partition access selection.

Definition at line 850 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.19 Sd_Ip_MmcBootPartitionType

enum [Sd_Ip_MmcBootPartitionType](#)

MMC card boot partition selection.

Definition at line 863 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.20 Sd_Ip_MmcBootSpeedType

enum [Sd_Ip_MmcBootSpeedType](#)

MMC card boot mode selection.

Definition at line 873 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.21 Sd_Ip_MmcBootWidthType

enum [Sd_Ip_MmcBootWidthType](#)

MMC card boot width selection.

Definition at line 881 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.22 Sd_Ip_MmcBusWidthType

enum [Sd_Ip_MmcBusWidthType](#)

MMC card bus width selection.

Definition at line 901 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.23 Sd_Ip_MmcCardType

enum [Sd_Ip_MmcCardType](#)

MMC card speed supports.

Definition at line 911 of file [Sd_Emmc_Ip_Types.h](#).

6.2.5.24 Sd_Ip_MmcSpeedModeType

enum [Sd_Ip_MmcSpeedModeType](#)

MMC card speed supports.

Definition at line 924 of file [Sd_Emmc_Ip_Types.h](#).

6.3 uSDHC

6.3.1 Detailed Description

Data Structures

- struct [Usdhc_Ip_aDma2DescriptorType](#)
Definition the ADMA2 descriptor structure. [More...](#)
- struct [Usdhc_Ip_BootConfigType](#)
The boot config for uSDHC. [More...](#)
- struct [Usdhc_Ip_TransferConfigType](#)
Card transfer configuration. [More...](#)
- struct [Usdhc_Ip_ConfigType](#)
Data structure to initialize the uSDHC. [More...](#)
- struct [Usdhc_Ip_DataType](#)
Card data descriptor. [More...](#)
- struct [Usdhc_Ip_CommandType](#)
Card command descriptor. [More...](#)
- struct [Usdhc_Ip_TransferType](#)
Transfer state. [More...](#)
- struct [Usdhc_Ip_ControllerCapabilityType](#)
uSDHC capability information. [More...](#)
- struct [Usdhc_Ip_StateType](#)
uSDHC current state. [More...](#)

Macros

- `#define USDHC_ALIGNMENT_SIZE`
uSDHC alignment size for different ADMA modes (4096U to make sure that aligned with both ADMA1's descriptor or ADMA2's descriptor)
- `#define USDHC_MAX_BLOCK_COUNT`
Maximum block count can be set one time.
- `#define USDHC_COMMAND_ERROR_INT`
Definitions for composed values of interrupt flags.
- `#define USDHC_DATA_ERROR_INT`
- `#define USDHC_ERROR_INT`
- `#define USDHC_DATA_INT`
- `#define USDHC_COMMAND_INT`
- `#define USDHC_ADMA1_ADDRESS_ALIGN`
The alignment size for ADDRESS filed in ADMA1's descriptor.
- `#define USDHC_ADMA1_LENGTH_ALIGN`
The alignment size for LENGTH field in ADMA1's descriptor.
- `#define USDHC_ADMA1_DESCRIPTOR_LENGTH`
Definition the length in bytes of ADMA1's descriptor.
- `#define USDHC_ADMA1_DESCRIPTOR_ADDR_SHIFT`
The bit shift for ADDRESS field in ADMA1's descriptor.
- `#define USDHC_ADMA1_DESCRIPTOR_ADDR_MASK`
The bit mask for ADDRESS field in ADMA1's descriptor.
- `#define USDHC_ADMA1_DESCRIPTOR_LENGTH_SHIFT`
The bit shift for LENGTH filed in ADMA1's descriptor.
- `#define USDHC_ADMA1_DESCRIPTOR_LENGTH_MASK`
The mask for LENGTH field in ADMA1's descriptor.
- `#define USDHC_ADMA1_DESCRIPTOR_MAX_LENGTH_PER_ENTRY`
The max value of LENGTH field in ADMA1's descriptor Because the address data on each table is 4KB align, the maximum length should be divisible by 4096.
- `#define USDHC_ADMA1_DESCRIPTOR_VALID_MASK`
The mask for the control/status fields in ADMA1 descriptor.
- `#define USDHC_ADMA2_ADDRESS_ALIGN`
The alignment size for ADDRESS field in ADMA2's descriptor.
- `#define USDHC_ADMA2_LENGTH_ALIGN`
The alignment size for LENGTH filed in ADMA2's descriptor.
- `#define USDHC_ADMA2_DESCRIPTOR_LENGTH`
Definition the length in bytes of ADMA2's descriptor.
- `#define USDHC_ADMA2_DESCRIPTOR_LENGTH_SHIFT`
The bit shift for LENGTH field in ADMA2's descriptor.
- `#define USDHC_ADMA2_DESCRIPTOR_LENGTH_MASK`
The bit mask for LENGTH field in ADMA2's descriptor.
- `#define USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY`
The max value of LENGTH field in ADMA2's descriptor Because the address data on each table is 32-bit boundary, the maximum length should be divisible by 4.
- `#define USDHC_ADMA2_DESCRIPTOR_VALID_MASK`
ADMA2 descriptor control and status masks.

- `#define USDHC_DATA0_LINE_LEVEL`
Definitions some macros to check signal level in DATx pins.
- `#define USDHC_DATA1_LINE_LEVEL`
- `#define USDHC_DATA2_LINE_LEVEL`
- `#define USDHC_DATA3_LINE_LEVEL`
- `#define USDHC_DATA4_LINE_LEVEL`
- `#define USDHC_DATA5_LINE_LEVEL`
- `#define USDHC_DATA6_LINE_LEVEL`
- `#define USDHC_DATA7_LINE_LEVEL`
- `#define USDHC_HOST_CTRL_CAP_SUPPORT_4BIT`
Flags to check if uSDHC supports 4 bit or 8 bit mode.
- `#define USDHC_RESPONSE_TYPE_R2_MASK`
Mask and shift macro to calculate the command response in case response type is R2.
- `#define USDHC_DEFAULT_ADMA2_TABLE_SIZE`
Default adma table size value for ADMA2.
- `#define USDHC_ENABLE_DMA_FLAG`
Transfer flags.
- `#define USDHC_ENABLE_BLOCK_COUNT_FLAG`
- `#define USDHC_ENABLE_AUTO_CMD12_FLAG`
- `#define USDHC_DATA_READ_FLAG`
- `#define USDHC_MULTIPLE_BLOCK_FLAG`
- `#define USDHC_CMD_TYPE_ABORT_FLAG`
- `#define USDHC_RESPONSE_LENGTH136_FLAG`
- `#define USDHC_RESPONSE_LENGTH48_FLAG`
- `#define USDHC_RESPONSE_LENGTH48_BUSY_FLAG`
- `#define USDHC_ENABLE_CRC_CHECK_FLAG`
- `#define USDHC_ENABLE_INDEX_CHECK_FLAG`
- `#define USDHC_DATA_PRESENT_FLAG`
- `#define USDHC_DMA_MODE_NO_DMA`
- `#define USDHC_DMA_MODE_ADMA1`
- `#define USDHC_DMA_MODE_ADMA2`

Types Reference

- `typedef uint32 Usdhc_Ip_aDma1DescriptorType`
Define the adma1 descriptor structure.
- `typedef uint32 Usdhc_Ip_DmaModeType`
DMA mode.
- `typedef void(* Usdhc_Ip_CallbackType) (uint32 instance, uint32 status)`
Callback function for uSDHC.

Enum Reference

- enum [Usdhc_Ip_StatusType](#)
Definition the USDHC status.
- enum [Usdhc_Ip_DataBusWidthType](#)
Data transfer width.
- enum [Usdhc_Ip_EndianModeType](#)
Endian mode.
- enum [Usdhc_Ip_CommandTypeType](#)
The command type.
- enum [Usdhc_Ip_CardControlType](#)
Card control masks.
- enum [Usdhc_Ip_ResponseType](#)
The command response type.
- enum [Usdhc_Ip_MmcBootFlagType](#)
Enable/Disable mmc boot flag masks.
- enum [Usdhc_Ip_BootModeType](#)
The boot mode for uSDHC.

Variables

- USDHC_Type *const [Usdhc_Ip_BaseAddr](#) [USDHC_INSTANCE_COUNT]
Usdhc base pointer array.
- uint32 [attribute](#)
- const uint32 * [address](#)
- uint32 [ackTimeout](#)
- [Usdhc_Ip_BootModeType](#) [bootMode](#)
- uint32 [stopBlockGapCount](#)
- [Usdhc_Ip_DataBusWidthType](#) [bootBusWidth](#)
- boolean [enableDDR](#)
- uint32 [bootFlags](#)
- uint32 [dataBlockSize](#)
- uint32 [dataBlockCount](#)
- uint32 [commandArgument](#)
- uint32 [commandIndex](#)
- uint32 [cmdReg](#)
- uint32 [mixCtrlReg](#)
- [Usdhc_Ip_CallbackType](#) [cardInsertCallback](#)
- [Usdhc_Ip_CallbackType](#) [cardRemoveCallback](#)
- [Usdhc_Ip_CallbackType](#) [transferCompleteCallback](#)
- [Usdhc_Ip_CallbackType](#) [transferErrorCallback](#)
- [Usdhc_Ip_CallbackType](#) [blockGapCallback](#)
- [Usdhc_Ip_CallbackType](#) [cardIntCallback](#)
- boolean [enableAutoCMD12](#)
- uint32 [blockSize](#)
- uint32 [blockCount](#)
- uint32 * [rxData](#)

- `const uint32 * txData`
- `uint32 index`
- `uint32 argument`
- `Usdhc_Ip_CommandType type`
- `Usdhc_Ip_ResponseType responseType`
- `uint32 response [4U]`
- `Usdhc_Ip_DataType * data`
- `Usdhc_Ip_CommandType * command`
- `uint32 maxBlockLength`
- `uint32 maxBlockCount`
- `uint32 flags`
- `uint32 * admaTable`
- `uint32 admaTableSize`
- `Usdhc_Ip_DataType *volatile data`
- `Usdhc_Ip_CommandType *volatile command`
- `volatile uint32 transferredWords`
- `Usdhc_Ip_CallbackType cardInsertCallback`
- `Usdhc_Ip_CallbackType cardRemoveCallback`
- `Usdhc_Ip_CallbackType transferCompleteCallback`
- `Usdhc_Ip_CallbackType transferErrorCallback`
- `Usdhc_Ip_CallbackType blockGapCallback`
- `Usdhc_Ip_CallbackType cardIntCallback`
- `volatile uint16 u16TransferComplete`

6.3.2 Data Structure Documentation

6.3.2.1 struct Usdhc_Ip_aDma2DescriptorType

Definition the ADMA2 descriptor structure.

Definition at line 298 of file [Usdhc_Ip_Types.h](#).

6.3.2.2 struct Usdhc_Ip_BootConfigType

The boot config for uSDHC.

This type is used to configure boot mode.

Definition at line 404 of file [Usdhc_Ip_Types.h](#).

6.3.2.3 struct Usdhc_Ip_TransferConfigType

Card transfer configuration.

Define structure to configure the transfer-related command index/argument/flags and data block size/data block numbers. This structure needs to be filled each time a command is sent to the card.

Definition at line 446 of file [Usdhc_Ip_Types.h](#).

6.3.2.4 struct Usdhc_Ip_ConfigType

Data structure to initialize the uSDHC.

Definition at line 458 of file [Usdhc_Ip_Types.h](#).

6.3.2.5 struct Usdhc_Ip_DataType

Card data descriptor.

Define structure to contain data-related attribute.

Definition at line 484 of file [Usdhc_Ip_Types.h](#).

6.3.2.6 struct Usdhc_Ip_CommandType

Card command descriptor.

Define card command-related attribute.

Definition at line 500 of file [Usdhc_Ip_Types.h](#).

6.3.2.7 struct Usdhc_Ip_TransferType

Transfer state.

Definition at line 512 of file [Usdhc_Ip_Types.h](#).

6.3.2.8 struct Usdhc_Ip_ControllerCapabilityType

uSDHC capability information.

Define structure to save the capability information of uSDHC.

Definition at line 523 of file [Usdhc_Ip_Types.h](#).

6.3.2.9 struct Usdhc_Ip_StateType

uSDHC current state.

Define structure to save the current state of uSDHC.

Definition at line 539 of file [Usdhc_Ip_Types.h](#).

6.3.3 Macro Definition Documentation

6.3.3.1 USDHC_ALIGNMENT_SIZE

```
#define USDHC_ALIGNMENT_SIZE
```

uSDHC alignment size for different ADMA modes (4096U to make sure that aligned with both ADMA1's descriptor or ADMA2's descriptor)

Definition at line 134 of file [Usdhc_Ip.h](#).

6.3.3.2 USDHC_MAX_BLOCK_COUNT

```
#define USDHC_MAX_BLOCK_COUNT
```

Maximum block count can be set one time.

Definition at line 94 of file [Usdhc_Ip_Types.h](#).

6.3.3.3 USDHC_COMMAND_ERROR_INT

```
#define USDHC_COMMAND_ERROR_INT
```

Definitions for composed values of interrupt flags.

< Command error Data error

Definition at line 103 of file [Usdhc_Ip_Types.h](#).

6.3.3.4 USDHC_DATA_ERROR_INT

```
#define USDHC_DATA_ERROR_INT
```

All error

Definition at line 109 of file [Usdhc_Ip_Types.h](#).

6.3.3.5 USDHC_ERROR_INT

```
#define USDHC_ERROR_INT
```

Data interrupts

Definition at line 114 of file [Usdhc_Ip_Types.h](#).

6.3.3.6 USDHC_DATA_INT

```
#define USDHC_DATA_INT
```

Command interrupts

Definition at line 122 of file [Usdhc_Ip_Types.h](#).

6.3.3.7 USDHC_COMMAND_INT

```
#define USDHC_COMMAND_INT
```

Card detection interrupts

Definition at line 126 of file [Usdhc_Ip_Types.h](#).

6.3.3.8 USDHC_ADMA1_ADDRESS_ALIGN

```
#define USDHC_ADMA1_ADDRESS_ALIGN
```

The alignment size for ADDRESS field in ADMA1's descriptor.

Definition at line 131 of file [Usdhc_Ip_Types.h](#).

6.3.3.9 USDHC_ADMA1_LENGTH_ALIGN

```
#define USDHC_ADMA1_LENGTH_ALIGN
```

The alignment size for LENGTH field in ADMA1's descriptor.

Definition at line 133 of file [Usdhc_Ip_Types.h](#).

6.3.3.10 USDHC_ADMA1_DESCRIPTOR_LENGTH

```
#define USDHC_ADMA1_DESCRIPTOR_LENGTH
```

Definition the length in bytes of ADMA1's descriptor.

Definition at line 136 of file [Usdhc_Ip_Types.h](#).

6.3.3.11 USDHC_ADMA1_DESCRIPTOR_ADDR_SHIFT

```
#define USDHC_ADMA1_DESCRIPTOR_ADDR_SHIFT
```

The bit shift for ADDRESS field in ADMA1's descriptor.

Definition at line 138 of file [Usdhc_Ip_Types.h](#).

6.3.3.12 USDHC_ADMA1_DESCRIPTOR_ADDR_MASK

```
#define USDHC_ADMA1_DESCRIPTOR_ADDR_MASK
```

The bit mask for ADDRESS field in ADMA1's descriptor.

Definition at line 140 of file [Usdhc_Ip_Types.h](#).

6.3.3.13 USDHC_ADMA1_DESCRIPTOR_LENGTH_SHIFT

```
#define USDHC_ADMA1_DESCRIPTOR_LENGTH_SHIFT
```

The bit shift for LENGTH field in ADMA1's descriptor.

Definition at line 142 of file [Usdhc_Ip_Types.h](#).

6.3.3.14 USDHC_ADMA1_DESCRIPTOR_LENGTH_MASK

```
#define USDHC_ADMA1_DESCRIPTOR_LENGTH_MASK
```

The mask for LENGTH field in ADMA1's descriptor.

Definition at line 144 of file [Usdhc_Ip_Types.h](#).

6.3.3.15 USDHC_ADMA1_DESCRIPTOR_MAX_LENGTH_PER_ENTRY

```
#define USDHC_ADMA1_DESCRIPTOR_MAX_LENGTH_PER_ENTRY
```

The max value of LENGTH field in ADMA1's descriptor Because the address data on each table is 4KB align, the maximum length should be divisible by 4096.

Definition at line 149 of file [Usdhc_Ip_Types.h](#).

6.3.3.16 USDHC_ADMA1_DESCRIPTOR_VALID_MASK

```
#define USDHC_ADMA1_DESCRIPTOR_VALID_MASK
```

The mask for the control/status fields in ADMA1 descriptor.

Definition at line 152 of file [Usdhc_Ip_Types.h](#).

6.3.3.17 USDHC_ADMA2_ADDRESS_ALIGN

```
#define USDHC_ADMA2_ADDRESS_ALIGN
```

The alignment size for ADDRESS field in ADMA2's descriptor.

Definition at line 163 of file [Usdhc_Ip_Types.h](#).

6.3.3.18 USDHC_ADMA2_LENGTH_ALIGN

```
#define USDHC_ADMA2_LENGTH_ALIGN
```

The alignment size for LENGTH field in ADMA2's descriptor.

Definition at line 165 of file [Usdhc_Ip_Types.h](#).

6.3.3.19 USDHC_ADMA2_DESCRIPTOR_LENGTH

```
#define USDHC_ADMA2_DESCRIPTOR_LENGTH
```

Definition the length in bytes of ADMA2's descriptor.

Definition at line 168 of file [Usdhc_Ip_Types.h](#).

6.3.3.20 USDHC_ADMA2_DESCRIPTOR_LENGTH_SHIFT

```
#define USDHC_ADMA2_DESCRIPTOR_LENGTH_SHIFT
```

The bit shift for LENGTH field in ADMA2's descriptor.

Definition at line 170 of file [Usdhc_Ip_Types.h](#).

6.3.3.21 USDHC_ADMA2_DESCRIPTOR_LENGTH_MASK

```
#define USDHC_ADMA2_DESCRIPTOR_LENGTH_MASK
```

The bit mask for LENGTH field in ADMA2's descriptor.

Definition at line 172 of file [Usdhc_Ip_Types.h](#).

6.3.3.22 USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY

```
#define USDHC_ADMA2_DESCRIPTOR_MAX_LENGTH_PER_ENTRY
```

The max value of LENGTH field in ADMA2's descriptor Because the address data on each table is 32-bit boundary, the maximum length should be divisible by 4.

Definition at line 177 of file [Usdhc_Ip_Types.h](#).

6.3.3.23 USDHC_ADMA2_DESCRIPTOR_VALID_MASK

```
#define USDHC_ADMA2_DESCRIPTOR_VALID_MASK
```

ADMA2 descriptor control and status masks.

Definition at line 180 of file [Usdhc_Ip_Types.h](#).

6.3.3.24 USDHC_DATA0_LINE_LEVEL

```
#define USDHC_DATA0_LINE_LEVEL
```

Definitions some macros to check signal level in DATx pins.

Data0 line signal level

Definition at line 191 of file [Usdhc_Ip_Types.h](#).

6.3.3.25 USDHC_DATA1_LINE_LEVEL

```
#define USDHC_DATA1_LINE_LEVEL
```

Data1 line signal level

Definition at line 192 of file [Usdhc_Ip_Types.h](#).

6.3.3.26 USDHC_DATA2_LINE_LEVEL

```
#define USDHC_DATA2_LINE_LEVEL
```

Data2 line signal level

Definition at line 193 of file [Usdhc_Ip_Types.h](#).

6.3.3.27 USDHC_DATA3_LINE_LEVEL

```
#define USDHC_DATA3_LINE_LEVEL
```

Data3 line signal level

Definition at line 194 of file [Usdhc_Ip_Types.h](#).

6.3.3.28 USDHC_DATA4_LINE_LEVEL

```
#define USDHC_DATA4_LINE_LEVEL
```

Data4 line signal level

Definition at line 195 of file [Usdhc_Ip_Types.h](#).

6.3.3.29 USDHC_DATA5_LINE_LEVEL

```
#define USDHC_DATA5_LINE_LEVEL
```

Data5 line signal level

Definition at line 196 of file [Usdhc_Ip_Types.h](#).

6.3.3.30 USDHC_DATA6_LINE_LEVEL

```
#define USDHC_DATA6_LINE_LEVEL
```

Data6 line signal level

Definition at line 197 of file [Usdhc_Ip_Types.h](#).

6.3.3.31 USDHC_DATA7_LINE_LEVEL

```
#define USDHC_DATA7_LINE_LEVEL
```

Data7 line signal level

Definition at line 198 of file [Usdhc_Ip_Types.h](#).

6.3.3.32 USDHC_HOST_CTRL_CAP_SUPPORT_4BIT

```
#define USDHC_HOST_CTRL_CAP_SUPPORT_4BIT
```

Flags to check if uSDHC supports 4 bit or 8 bit mode.

Definition at line 201 of file [Usdhc_Ip_Types.h](#).

6.3.3.33 USDHC_RESPONSE_TYPE_R2_MASK

```
#define USDHC_RESPONSE_TYPE_R2_MASK
```

Mask and shift macro to calculate the command response in case response type is R2.

Definition at line 205 of file [Usdhc_Ip_Types.h](#).

6.3.3.34 USDHC_DEFAULT_ADMA2_TABLE_SIZE

```
#define USDHC_DEFAULT_ADMA2_TABLE_SIZE
```

Default adma table size value for ADMA2.

Definition at line 209 of file [Usdhc_Ip_Types.h](#).

6.3.3.35 USDHC_ENABLE_DMA_FLAG

```
#define USDHC_ENABLE_DMA_FLAG
```

Transfer flags.

Enable DMA

Definition at line 212 of file [Usdhc_Ip_Types.h](#).

6.3.3.36 USDHC_ENABLE_BLOCK_COUNT_FLAG

```
#define USDHC_ENABLE_BLOCK_COUNT_FLAG
```

Enable block count

Definition at line 213 of file [Usdhc_Ip_Types.h](#).

6.3.3.37 USDHC_ENABLE_AUTO_CMD12_FLAG

```
#define USDHC_ENABLE_AUTO_CMD12_FLAG
```

Enable auto CMD12

Definition at line 214 of file [Usdhc_Ip_Types.h](#).

6.3.3.38 USDHC_DATA_READ_FLAG

```
#define USDHC_DATA_READ_FLAG
```

Enable data read

Definition at line 215 of file [Usdhc_Ip_Types.h](#).

6.3.3.39 USDHC_MULTIPLE_BLOCK_FLAG

```
#define USDHC_MULTIPLE_BLOCK_FLAG
```

Multiple block data read/write

Definition at line 216 of file [Usdhc_Ip_Types.h](#).

6.3.3.40 USDHC_CMD_TYPE_ABORT_FLAG

```
#define USDHC_CMD_TYPE_ABORT_FLAG
```

Abort command

Definition at line 218 of file [Usdhc_Ip_Types.h](#).

6.3.3.41 USDHC_RESPONSE_LENGTH136_FLAG

```
#define USDHC_RESPONSE_LENGTH136_FLAG
```

136 bit response length

Definition at line 220 of file [Usdhc_Ip_Types.h](#).

6.3.3.42 USDHC_RESPONSE_LENGTH48_FLAG

```
#define USDHC_RESPONSE_LENGTH48_FLAG
```

48 bit response length

Definition at line 221 of file [Usdhc_Ip_Types.h](#).

6.3.3.43 USDHC_RESPONSE_LENGTH48_BUSY_FLAG

```
#define USDHC_RESPONSE_LENGTH48_BUSY_FLAG
```

48 bit response length with busy status

Definition at line 222 of file [Usdhc_Ip_Types.h](#).

6.3.3.44 USDHC_ENABLE_CRC_CHECK_FLAG

```
#define USDHC_ENABLE_CRC_CHECK_FLAG
```

Enable CRC check

Definition at line 224 of file [Usdhc_Ip_Types.h](#).

6.3.3.45 USDHC_ENABLE_INDEX_CHECK_FLAG

```
#define USDHC_ENABLE_INDEX_CHECK_FLAG
```

Enable index check

Definition at line 225 of file [Usdhc_Ip_Types.h](#).

6.3.3.46 USDHC_DATA_PRESENT_FLAG

```
#define USDHC_DATA_PRESENT_FLAG
```

Data present flag

Definition at line 226 of file [Usdhc_Ip_Types.h](#).

6.3.3.47 USDHC_DMA_MODE_NO_DMA

```
#define USDHC_DMA_MODE_NO_DMA
```

No DMA, use interrupt or polling method for transferring

Definition at line 326 of file [Usdhc_Ip_Types.h](#).

6.3.3.48 USDHC_DMA_MODE_ADMA1

```
#define USDHC_DMA_MODE_ADMA1
```

ADMA1 is selected for transferring

Definition at line 327 of file [Usdhc_Ip_Types.h](#).

6.3.3.49 USDHC_DMA_MODE_ADMA2

```
#define USDHC_DMA_MODE_ADMA2
```

ADMA2 is selected for transferring

Definition at line 328 of file [Usdhc_Ip_Types.h](#).

6.3.4 Types Reference**6.3.4.1 Usdhc_Ip_aDma1DescriptorType**

```
typedef uint32 Usdhc_Ip_aDma1DescriptorType
```

Define the adma1 descriptor structure.

Definition at line 272 of file [Usdhc_Ip_Types.h](#).

6.3.4.2 Usdhc_Ip_DmaModeType

```
typedef uint32 Usdhc_Ip_DmaModeType
```

DMA mode.

Definition at line 325 of file [Usdhc_Ip_Types.h](#).

6.3.4.3 Usdhc_Ip_CallbackType

```
typedef void(* Usdhc_Ip_CallbackType) (uint32 instance, uint32 status)
```

Callback function for uSDHC.

Parameters

<i>instance</i>	uSDHC number instance invoking this function
<i>status</i>	This parameter is used for transfer complete callback function. It indicates the status of a transfer. In case of success, value of status is STATUS_SUCCESS and in case of failure, this value will contain the interrupt flag of uSDHC module.
<i>param</i>	The user data that registered before by InstallCallback functions

Definition at line 439 of file [Usdhc_Ip_Types.h](#).

6.3.5 Enum Reference

6.3.5.1 Usdhc_Ip_StatusType

enum [Usdhc_Ip_StatusType](#)

Definition the USDHC status.

Enumerator

STATUS_USDHC_IP_SUCCESS	Operation success status
STATUS_USDHC_IP_PENDING	Operation pending status
STATUS_USDHC_IP_ERROR	Operation error status
STATUS_USDHC_IP_BUSY	Operation busy status
STATUS_USDHC_IP_TIMEOUT	Operation timeout status
STATUS_USDHC_IP_OUT_OF_RANGE	The size of data to be sent is larger than maximum size of ADMA table
STATUS_USDHC_IP_PREPARE_ADMA_FAILED	Failed to prepare the ADMA table

Definition at line 238 of file [Usdhc_Ip_Types.h](#).

6.3.5.2 Usdhc_Ip_DataBusWidthType

enum [Usdhc_Ip_DataBusWidthType](#)

Data transfer width.

Enumerator

USDHC_DATA_BUS_WIDTH_1BIT	1-bit mode
USDHC_DATA_BUS_WIDTH_4BIT	4-bit mode
USDHC_DATA_BUS_WIDTH_8BIT	8-bit mode

Definition at line 307 of file [Usdhc_Ip_Types.h](#).

6.3.5.3 Usdhc_Ip_EndianModeType

enum [Usdhc_Ip_EndianModeType](#)

Endian mode.

Enumerator

USDHC_ENDIAN_MODE_BIG	Big endian mode
USDHC_ENDIAN_MODE_HALF_WORD_BIG	Half word big endian mode
USDHC_ENDIAN_MODE_LITTLE	Little endian mode

Definition at line 316 of file [Usdhc_Ip_Types.h](#).

6.3.5.4 Usdhc_Ip_CommandTypeType

enum [Usdhc_Ip_CommandTypeType](#)

The command type.

Enumerator

USDHC_COMMAND_TYPE_NORMAL	Normal command
USDHC_COMMAND_TYPE_SUSPEND	Suspend command
USDHC_COMMAND_TYPE_RESUME	Resume command
USDHC_COMMAND_TYPE_ABORT	Abort command

Definition at line 333 of file [Usdhc_Ip_Types.h](#).

6.3.5.5 Usdhc_Ip_CardControlType

enum [Usdhc_Ip_CardControlType](#)

Card control masks.

Enumerator

USDHC_WAKEUP_ON_CARD_INT	Wakeup on card interrupt
USDHC_WAKEUP_ON_CARD_INSERT	Wakeup on card insertion
USDHC_WAKEUP_ON_CARD_REMOVE	Wakeup on card removal
USDHC_STOP_AT_BLOCK_GAP	Stop at block gap request
USDHC_READ_WAIT_CONTROL	Read wait control
USDHC_INT_AT_BLOCK_GAP	Interrupt at block gap
USDHC_NON_EXACT_BLOCK_READ	Non exact block read
USDHC_CONTINUE_REQUEST	Continue request

Definition at line 343 of file [Usdhc_Ip_Types.h](#).

6.3.5.6 Usdhc_Ip_ResponseType

```
enum Usdhc_Ip_ResponseType
```

The command response type.

Define the command response type from card to host controller.

Enumerator

USDHC_RESPONSE_TYPE_NONE	Response type: none
USDHC_RESPONSE_TYPE_R1	Response type: R1
USDHC_RESPONSE_TYPE_R1b	Response type: R1b
USDHC_RESPONSE_TYPE_R2	Response type: R2
USDHC_RESPONSE_TYPE_R3	Response type: R3
USDHC_RESPONSE_TYPE_R4	Response type: R4
USDHC_RESPONSE_TYPE_R5	Response type: R5
USDHC_RESPONSE_TYPE_R5b	Response type: R5b
USDHC_RESPONSE_TYPE_R6	Response type: R6
USDHC_RESPONSE_TYPE_R7	Response type: R7

Definition at line 360 of file [Usdhc_Ip_Types.h](#).

6.3.5.7 Usdhc_Ip_MmcBootFlagType

enum [Usdhc_Ip_MmcBootFlagType](#)

Enable/Disable mmc boot flag masks.

Enumerator

USDHC_ENABLE_BOOT_ACK_FLAG	Enable the boot ACK
USDHC_DISABLE_BOOT_ACK_FLAG	Disable the boot ACK
USDHC_ENABLE_FAST_BOOT_FLAG	Enable the fast boot
USDHC_DISABLE_FAST_BOOT_FLAG	Disable the fast boot
USDHC_DISABLE_TIME_OUT_FLAG	Disable check timeout
USDHC_ENABLE_TIME_OUT_FLAG	Enable check timeout
USDHC_ENABLE_BOOT_STOP_AT_BLK_GAP_FLAG	Enable the automatic stop at the block gap
USDHC_DISABLE_BOOT_STOP_AT_BLK_GAP_FLAG	Disable the automatic stop at the block gap

Definition at line 376 of file [Usdhc_Ip_Types.h](#).

6.3.5.8 Usdhc_Ip_BootModeType

enum [Usdhc_Ip_BootModeType](#)

The boot mode for uSDHC.

This enumeration is used to configure boot mode.

Enumerator

USDHC_NORMAL_MMC_BOOT	Normal MMC boot
USDHC_ALTER_MMC_BOOT	Alternative MMC boot

Definition at line 393 of file [Usdhc_Ip_Types.h](#).

6.3.6 Variable Documentation

6.3.6.1 Usdhc_Ip_BaseAddr

```
USDHC_Type* const Usdhc_Ip_BaseAddr[USDHC_INSTANCE_COUNT] [extern]
```

Usdhc base pointer array.

6.3.6.2 attribute

```
uint32 attribute
```

The control and status field

Definition at line 300 of file [Usdhc_Ip_Types.h](#).

6.3.6.3 address

```
const uint32* address
```

The address field

Definition at line 301 of file [Usdhc_Ip_Types.h](#).

6.3.6.4 ackTimeout

```
uint32 ackTimeout
```

Sets the timeout value for the boot ACK. NOTE: In register configuration, real timeout is:

ackTimeout	S32V234	S32G274A	S32R45
0	SDCLK x 2 [^] 13	SDCLK x 2 [^] 32	SDCLK x 2 [^] 14
1	SDCLK x 2 [^] 14	SDCLK x 2 [^] 33	SDCLK x 2 [^] 15
2	SDCLK x 2 [^] 15	SDCLK x 2 [^] 18	SDCLK x 2 [^] 16
3	SDCLK x 2 [^] 16	SDCLK x 2 [^] 19	SDCLK x 2 [^] 17
4	SDCLK x 2 [^] 17	SDCLK x 2 [^] 20	SDCLK x 2 [^] 18
5	SDCLK x 2 [^] 18	SDCLK x 2 [^] 21	SDCLK x 2 [^] 19
6	SDCLK x 2 [^] 19	SDCLK x 2 [^] 22	SDCLK x 2 [^] 20
7	SDCLK x 2 [^] 20	SDCLK x 2 [^] 23	SDCLK x 2 [^] 21
...

Definition at line 406 of file [Usdhc_Ip_Types.h](#).

6.3.6.5 bootMode

[Usdhc_Ip_BootModeType](#) bootMode

Configures the boot mode.

Definition at line 423 of file [Usdhc_Ip_Types.h](#).

6.3.6.6 stopBlockGapCount

uint32 stopBlockGapCount

Configures the block count for boot operation to stop at block gap, BLK_CNT - BOOT_BLK_CNT

Definition at line 424 of file [Usdhc_Ip_Types.h](#).

6.3.6.7 bootBusWidth

[Usdhc_Ip_DataBusWidthType](#) bootBusWidth

Bus width selection for boot operation

Definition at line 425 of file [Usdhc_Ip_Types.h](#).

6.3.6.8 enableDDR

boolean enableDDR

Enable double data rate mode for boot operation

Definition at line 426 of file [Usdhc_Ip_Types.h](#).

6.3.6.9 bootFlags

uint32 bootFlags

Enable MMC boot function (a logical OR of "Usdhc_Ip_MmcBootFlagType")

Definition at line 427 of file [Usdhc_Ip_Types.h](#).

6.3.6.10 dataBlockSize

uint32 dataBlockSize

Data block size

Definition at line 448 of file [Usdhc_Ip_Types.h](#).

6.3.6.11 dataBlockCount

uint32 dataBlockCount

Data block count

Definition at line 449 of file [Usdhc_Ip_Types.h](#).

6.3.6.12 commandArgument

uint32 commandArgument

Command argument

Definition at line 450 of file [Usdhc_Ip_Types.h](#).

6.3.6.13 commandIndex

uint32 commandIndex

Command index

Definition at line 451 of file [Usdhc_Ip_Types.h](#).

6.3.6.14 cmdReg

uint32 cmdReg

Register values for CMD_XFR_TYP register

Definition at line 452 of file [Usdhc_Ip_Types.h](#).

6.3.6.15 mixCtrlReg

uint32 mixCtrlReg

Register values for MIX_CTRL register

Definition at line 453 of file [Usdhc_Ip_Types.h](#).

6.3.6.16 cardInsertCallback [1/2]

[Usdhc_Ip_CallbackType](#) cardInsertCallback

Card insertion callback function

Definition at line 470 of file [Usdhc_Ip_Types.h](#).

6.3.6.17 cardRemoveCallback [1/2]

`Usdhc_Ip_CallbackType` cardRemoveCallback

Card removal callback function

Definition at line 471 of file [Usdhc_Ip_Types.h](#).

6.3.6.18 transferCompleteCallback [1/2]

`Usdhc_Ip_CallbackType` transferCompleteCallback

Transfer complete callback function

Definition at line 472 of file [Usdhc_Ip_Types.h](#).

6.3.6.19 transferErrorCallback [1/2]

`Usdhc_Ip_CallbackType` transferErrorCallback

Transfer error callback function

Definition at line 473 of file [Usdhc_Ip_Types.h](#).

6.3.6.20 blockGapCallback [1/2]

`Usdhc_Ip_CallbackType` blockGapCallback

Block gap callback function

Definition at line 474 of file [Usdhc_Ip_Types.h](#).

6.3.6.21 cardIntCallback [1/2]

`Usdhc_Ip_CallbackType cardIntCallback`

Card interrupt callback function

Definition at line 475 of file [Usdhc_Ip_Types.h](#).

6.3.6.22 enableAutoCMD12

`boolean enableAutoCMD12`

Enable auto CMD12. If this field is false, user have to send a "CMD12" command to card after each time send/receive data to stop the data transfer. If this field is true, uSDHC will send "CMD12" command automatically after data transfer is end

Definition at line 486 of file [Usdhc_Ip_Types.h](#).

6.3.6.23 blockSize

`uint32 blockSize`

Block size for each data transfer

Definition at line 489 of file [Usdhc_Ip_Types.h](#).

6.3.6.24 blockCount

`uint32 blockCount`

Block count for each data transfer

Definition at line 490 of file [Usdhc_Ip_Types.h](#).

6.3.6.25 rxData

```
uint32* rxData
```

Buffer to store the read data

Definition at line [491](#) of file [Usdhc_Ip_Types.h](#).

6.3.6.26 txData

```
const uint32* txData
```

Data buffer to write

Definition at line [492](#) of file [Usdhc_Ip_Types.h](#).

6.3.6.27 index

```
uint32 index
```

Command index

Definition at line [502](#) of file [Usdhc_Ip_Types.h](#).

6.3.6.28 argument

```
uint32 argument
```

Command argument

Definition at line [503](#) of file [Usdhc_Ip_Types.h](#).

6.3.6.29 type

`Usdhc_Ip_CommandTypeType` type

Command type

Definition at line 504 of file [Usdhc_Ip_Types.h](#).

6.3.6.30 responseType

`Usdhc_Ip_ResponseType` responseType

Command response type

Definition at line 505 of file [Usdhc_Ip_Types.h](#).

6.3.6.31 response

`uint32 response[4U]`

Response for this command

Definition at line 506 of file [Usdhc_Ip_Types.h](#).

6.3.6.32 data [1/2]

`Usdhc_Ip_DataType* data`

Data to transfer

Definition at line 514 of file [Usdhc_Ip_Types.h](#).

6.3.6.33 command [1/2]

`Usdhc_Ip_CommandType* command`

Command to send

Definition at line 515 of file [Usdhc_Ip_Types.h](#).

6.3.6.34 maxBlockLength

`uint32 maxBlockLength`

Maximum block length united as byte

Definition at line 529 of file [Usdhc_Ip_Types.h](#).

6.3.6.35 maxBlockCount

`uint32 maxBlockCount`

Maximum block count can be set one time

Definition at line 530 of file [Usdhc_Ip_Types.h](#).

6.3.6.36 flags

`uint32 flags`

Capability flags to indicate the support information

Definition at line 531 of file [Usdhc_Ip_Types.h](#).

6.3.6.37 admaTable

```
uint32* admaTable
```

ADMA table is used to transfer

Definition at line 541 of file [Usdhc_Ip_Types.h](#).

6.3.6.38 admaTableSize

```
uint32 admaTableSize
```

The size of ADMA table in

Definition at line 542 of file [Usdhc_Ip_Types.h](#).

6.3.6.39 data [2/2]

```
Usdhc_Ip_DataType* volatile data
```

Data to transfer

Definition at line 544 of file [Usdhc_Ip_Types.h](#).

6.3.6.40 command [2/2]

```
Usdhc_Ip_CommandType* volatile command
```

Command to send

Definition at line 545 of file [Usdhc_Ip_Types.h](#).

6.3.6.41 transferredWords

```
volatile uint32 transferredWords
```

Words transferred by DATAPORT way

Definition at line 548 of file [Usdhc_Ip_Types.h](#).

6.3.6.42 cardInsertCallback [2/2]

```
Usdhc_Ip_CallbackType cardInsertCallback
```

Card insertion callback function

Definition at line 550 of file [Usdhc_Ip_Types.h](#).

6.3.6.43 cardRemoveCallback [2/2]

```
Usdhc_Ip_CallbackType cardRemoveCallback
```

Card removal callback function

Definition at line 551 of file [Usdhc_Ip_Types.h](#).

6.3.6.44 transferCompleteCallback [2/2]

```
Usdhc_Ip_CallbackType transferCompleteCallback
```

Transfer complete callback function

Definition at line 552 of file [Usdhc_Ip_Types.h](#).

6.3.6.45 transferErrorCallback [2/2]

`Usdhc_Ip_CallbackType transferErrorCallback`

Transfer error callback function

Definition at line 553 of file [Usdhc_Ip_Types.h](#).

6.3.6.46 blockGapCallback [2/2]

`Usdhc_Ip_CallbackType blockGapCallback`

Block gap callback function

Definition at line 554 of file [Usdhc_Ip_Types.h](#).

6.3.6.47 cardIntCallback [2/2]

`Usdhc_Ip_CallbackType cardIntCallback`

Card interrupt callback function

Definition at line 555 of file [Usdhc_Ip_Types.h](#).

6.3.6.48 u16TransferComplete

`volatile uint16 u16TransferComplete`

Synchronization object for blocking transfer timeout condition

Definition at line 556 of file [Usdhc_Ip_Types.h](#).

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2023 NXP B.V.

