User Manual

for S32K3 OCOTP Driver

Document Number: UM34OCOTPASRR21-11 Rev0000R3.0.0 Rev. 1.0

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

This User Manual describes for One Time Programmable Memory (MEM OTP) Driver for S32K3. MEM OTP driver configuration parameters and deviations from the specification are described in Driver chapter of this document.

2.1 Supported Derivatives

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

- 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	Development Error Tracer
ECU	Electronic Control Unit
MCU	Micro Controller Unit
OS	Operating System
MSB	Most Significant Bit
N/A	Not Applicable
OCOTP	On-Chip One Time Programmable
MEM OTP	One time programmable memory
OTPC	One-time-programmable memory controller
HW	Hardware

2.5 Reference List

#	Title	Version
1	Reference Manual	S32M27x Reference Manual, Rev.2, Draft A, — 02/2023
2	Datasheet	S32M2xx Data Sheet, Rev. 2 RC — 12/2022

Chapter 3

Driver

3.1 Requirements

Deviations from requirements are described in Deviations from Requirements chapter of this document.

3.2 Driver Design Summary

MEM_OTP driver is a one-time-programmable memory controller (OTPC). It supports the execution of commands to manage the operation of the OTP memory. The main logic, the fuse block, consists of a set of mirror registers and the OTPC. The OTPC has 3 functions:

- At boot time, it reads the contents of the OTP memory(eFuse) into the mirror registers.
- It executes commands in response to the OTPC registers
- It performs tests on the OTP memory

There are 53 mirror registers, each register has 8 bits:

- 37 registers starting at address 0 to 36.

- 12 registers starting at address 64 ECID: unique sample identifier.
- 4 registers starting at address 96: temperature monitors area.

MEM_OTP driver is designed to provide APIs to implement the following functions:

- Programming and reading of 8-bits mirror registers and reading 8-bits eFuses.

The user performs programming with the eFuse byte and corresponding mirror registers by configuring the channels. Each channel includes:

- Channel name, channel index that is used to call parameters for the api in the MEM_OTP driver.
- An eFuse byte, includes information:
  - The name of eFuse.
  - The address of eFuse.

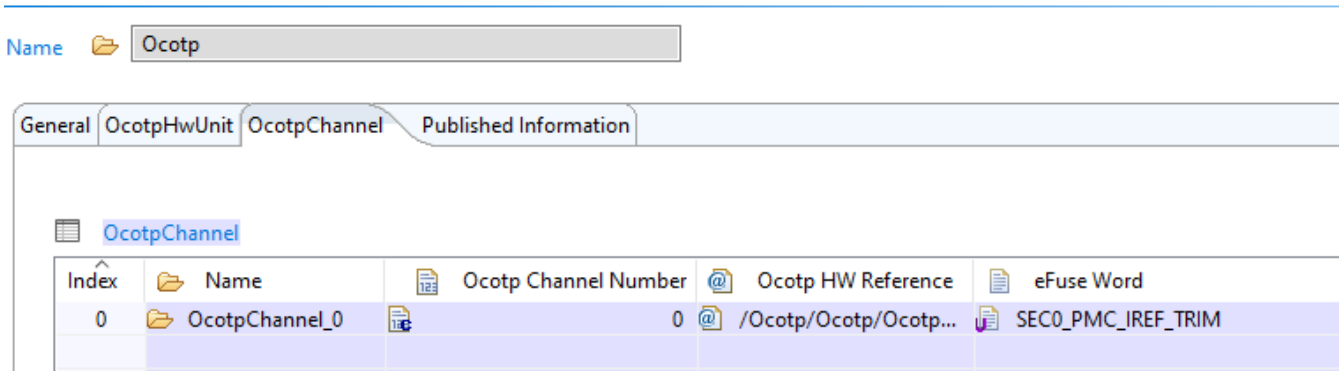

Figure 3.1 Channel configuration

3.3 Hardware Resources

MEM OTP Hardware Resource includes 53 mirror registers, each register has 8 bits:

- 37 registers starting at address 0 to 36.
- 12 registers starting at address 64 ECID: unique sample identifier.
- 4 registers starting at address 96: temperature monitors area.

3.4 Deviations from Requirements

The driver deviates from the OCOTP Driver software specification in someplaces. There are also some additional requirements (on top of requirements detailed in OCOTP Driver software specification) which need to be satisfied for correct operation.

- Deviations Status Column Description

Term	Definition
N/S	Not In Scope
N/F	Not Fully Implemented
N/I	Not Implemented

None.

3.5 Driver Limitations

- Before writing into Mirror and Otp register, need to write value 0x455C and 0xE80C into MODE_CONTROL register of AEC region. But this register is not showed in Reference Manual.
- Reference Manual does not mention to Efuse's address area nor to OTP's mirror registers
- In Mem_Otp_ReadOTPMem function, there are some registers that this RM does not show, such as: MR↩EF_TEST (address 0x138), MR_TST(0x13A), ADVCFG0 (0x12C), ADVCFG3 (0x132).

3.6 Driver usage and configuration tips

- Fuse Map Table
- Ocotp Timeout Supervision
- Configuration

3.6.1 Fuse Map Table

FuseMap is divided into 53 registers, each register is 8 bits. Each register contains information to configure a particular driver.

3.6.2 Ocotp Timeout Supervision

Ocotp will be in a busy state if it is in processes:

- Read from Mirror
- Write into Mirror
- Read from eFuse

  Therefore, some of the following APIs need to be configued Timeout for OCOTP to complete the process.

- Ocotp_ReadEFuse();
- Ocotp_ReadMirrorRegister();

- Ocotp_WriteMirrorRegister();

  User can refer to the configuration for Timeout as shown below.

Figure 3.2 Configuration for timeout

3.6.3 Configuration

A channel will contain some independent information:

- General information:
  - Ocotp Channel Number
  - Ocotp HW Reference
- An eFuse Word information:
  - eFuse Word : the eFuse byte address and corresponding Mirror Register address

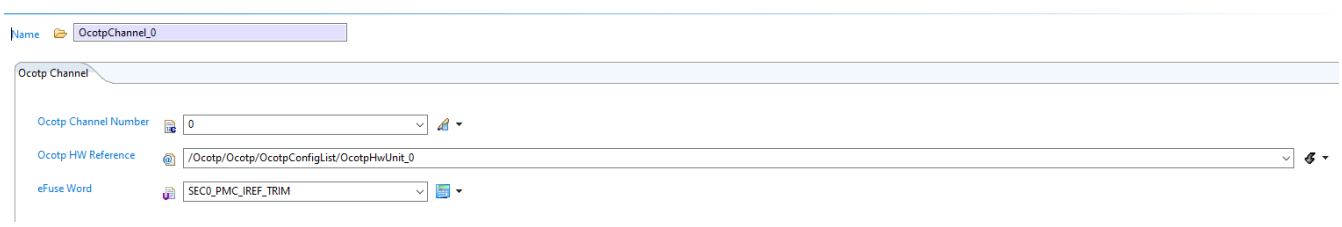

Figure 3.3 Channel configuration infomation

3.7 Runtime errors

- The driver supports runtime generation of the errors listed in the Table:

Function	Error code	Condition triggering the error
Ocotp_ReadEFuse()	OCOTP_E_TIMEOUT	(Ocotp Timeout Supervision Enabled == ON) and Ocotp is still busy when the timeout is over.

- The driver supports Transient faults generation of the errors listed in the Table.

Function	Error Code	Condition triggering the error
Ocotp_WriteMirrorRegister()	OCOTP_E_WRITE_FAILED	Write operation failed.(in the case of writing to a write-protected Mirror register).
Ocotp_ReadEFuse()	OCOTP_E_READ_FAILED	Read operation failed.
Ocotp_ReadMirrorRegister()	OCOTP_E_READ_FAILED	Read operation failed.

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 Ocotp
  - Container OcotpGeneral
    - Parameter OcotpDevErrorDetect
    - Parameter OcotpIpDevErrorDetect
    - Parameter OcotpEnableUserModeSupport
    - Parameter OcotpReadEFuseApi
    - Parameter OcotpReadMirrorRegisterApi
    - Parameter OcotpWriteMirrorRegisterApi
    - Parameter OcotpReadMirrorWithCommand
    - Parameter OcotpWriteMirrorWithCommand
    - Parameter OcotpGetStatusApi
    - Parameter OcotpTimeoutMethod
    - Parameter OcotpWriteTimeout
    - Parameter OcotpReadTimeout
  - Container OcotpConfigList
    - Container OcotpHwUnit
      - Parameter OcotpHardwareIndex
      - Parameter OcotpInstance
  - Container OcotpChannelList
    - Container OcotpChannel
      - Parameter OcotpChannelNumber
      - Parameter OcotpWord
      - Reference OcotpHwRef
  - Container CommonPublishedInformation
    - Parameter ArReleaseMajorVersion
    - Parameter ArReleaseMinorVersion
    - Parameter ArReleaseRevisionVersion
    - Parameter ModuleId
    - Parameter SwMajorVersion
    - Parameter SwMinorVersion
    - Parameter SwPatchVersion
    - Parameter VendorApiInfix
    - Parameter VendorId

4.1 Module Ocotp

Configuration of the Ocotp module.

Included containers:

- OcotpGeneral
- OcotpConfigList
- OcotpChannelList
- CommonPublishedInformation

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

4.2 Container OcotpGeneral

Container for general parameters of the ocotp 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 OcotpDevErrorDetect

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

4.4 Parameter OcotpIpDevErrorDetect

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

true: Ocotp error detect for IP layer is enabled.

false: Ocotp error detect for IP layer is 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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.5 Parameter OcotpEnableUserModeSupport

Ocotp module can run in user mode without any specific measures. The parameter is not used in the Ocotp implementation.

Property	Value
type	ECUC-BOOLEAN-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
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.6 Parameter OcotpReadEFuseApi

Compile switch to enable and disable the Ocotp_ReadEFuse 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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.7 Parameter OcotpReadMirrorRegisterApi

Compile switch to enable and disable the OcotpReadMirrorRegisterApi 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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.8 Parameter OcotpWriteMirrorRegisterApi

Compile switch to enable and disable the Ocotp_OcotpWriteMirrorRegister 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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.9 Parameter OcotpReadMirrorWithCommand

Enable Read Mirror With Command.

true: Read Mirror With Command.

false: Read Mirror directly.

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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.10 Parameter OcotpWriteMirrorWithCommand

Enable Write Mirror With Command.

true: Write Mirror With Command.

false: Write Mirror directly.

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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.11 Parameter OcotpGetStatusApi

Compile switch to enable and disable the Ocotp_GetStatus 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-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.12 Parameter OcotpTimeoutMethod

OcotpTimeoutMethod

Configures the timeout method.

Based on this selection a certain timeout method from OsIf will be used in the driver.

Note: If OSIF_COUNTER_SYSTEM or OSIF_COUNTER_CUSTOM are selected make sure the corresponding timer is enabled in OsIf General configuration.

Note: Implementation Specific Parameter.

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
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COU↩NTER_CUSTOM']

4.13 Parameter OcotpWriteTimeout

Timeout value for write eFuse operation.

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-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	0

4.14 Parameter OcotpReadTimeout

Timeout value for read eFuse operation.

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-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4294967295
min	0

4.15 Container OcotpConfigList

Container for runtime configuration parameters of the Ocotp driver.

Implementation Type: Ocotp_ConfigType.

Included subcontainers:

- OcotpHwUnit

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

4.16 Container OcotpHwUnit

Ocotp Hardware Unit.

Included subcontainers:

- None

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

4.17 Parameter OcotpHardwareIndex

Index of current hardware unit.

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

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.18 Parameter OcotpInstance

Instance of this word channel

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

4.19 Container OcotpChannelList

List of Otp channel.

Included subcontainers:

- OcotpChannel

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

4.20 Container OcotpChannel

Ocotp Channel Configuration.

Included subcontainers:

- None

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

4.21 Parameter OcotpChannelNumber

Index of current channel.

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

4.22 Parameter OcotpWord

Ocotp Fuse Word.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SEC0_PMC_IREF_TRIM
literals	['SEC0_PMC_IREF_TRIM', 'SEC0_PMC_VREF_TRIM', 'SEC0_PMC_↩ VDD_VDDC_VDIG15_TRIM', 'SEC0_OSC42M', 'SEC0_CAN_WUOSC_↩ TRIM', 'SEC0_CAN_OSC_TRIM', 'SEC0_CAN_EMETRIM', 'SEC0_LIN↩ _DPIMASK_LIW', 'SEC0_LIN_SETDISSCP_LIW', 'SEC0_LIN_CXPI_0', 'SEC0_LIN_CXPI_1', 'SEC0_CRC0_0', 'SEC0_CRC0_1', 'SEC1_PMC_I↩ REF_TRIM', 'SEC1_PMC_VREF_TRIM', 'SEC1_PMC_VDD_VDDC_V↩ DIG15_TRIM', 'SEC1_OSC42M', 'SEC1_LIN_DPIMASK_LIW', 'SEC1_L↩ IN_SETDISSCP_LIW', 'SEC1_LIN_CXPI_0', 'SEC1_LIN_CXPI_1', 'SE↩ C1_CRC1_0', 'SEC1_CRC1_1', 'SEC2_CAN_RX_TRIM', 'SEC2_CAN_I↩ REF_TRIM', 'SEC2_CAN_WUOSC_TRIM', 'SEC2_CAN_OSC_TRIM', 'S↩ EC2_CAN_TXD_ADJUST', 'SEC2_CAN_RXD_ADJUST', 'SEC2_CAN_↩ EMETRIM', 'SEC2_CRC2_0', 'SEC2_CRC2_1', 'SEC3_GDU_IRT1P0', 'S↩ EC3_GDU_IRT1P1', 'SEC3_GDU_IRT1P2', 'SEC3_CRC3_0', 'SEC3_CR↩ C3_1', 'ECID_0', 'ECID_1', 'ECID_2', 'ECID_3', 'ECID_4', 'ECID_5', 'EC↩ ID_6', 'ECID_7', 'ECID_8', 'ECID_9', 'ECID_10', 'ECID_11', 'TEMP_SE↩ NSOR_0', 'TEMP_SENSOR_1', 'TEMP_SENSOR_2', 'TEMP_SENSOR_3']

4.23 Reference OcotpHwRef

Select Ocotp Hardware Reference

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Ocotp/OcotpConfigList/OcotpHwUnit

4.24 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

4.25 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
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.26 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-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.27 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
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.28 Parameter ModuleId

Vendor specific: Module ID of this module.

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-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	255
max	255
min	255

4.29 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
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

4.30 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

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

4.31 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-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.32 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-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.33 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
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	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:

Chapter 6

Module Documentation

6.1 OCOTP

6.1.1 Detailed Description

Data Structures

- struct Ocotp_ChannelConfigType

  Ocotp channel configuration type. More...
- struct Ocotp_ConfigType

  Ocotp configuration type. More...
- struct Ocotp_StatusType

  Ocotp status type. More...

Macros

- #define OCOTP_MODULE_ID

  ID of module and instance.
- #define OCOTP_INSTANCE_ID
- #define OCOTP_INVALID_INDEX

  Invalid value of index and address.
- #define OCOTP_INVALID_ADDRESS
- #define OCOTP_E_PARAM_CONFIG

  Development error codes (passed to DET).
- #define OCOTP_E_PARAM_LENGTH
- #define OCOTP_E_PARAM_POINTER
- #define OCOTP_E_PARAM_CHANNEL
- #define OCOTP_E_PARAM_INSTANCE
- #define OCOTP_E_PARAM_MODE
- #define OCOTP_E_UNINIT
- #define OCOTP_E_ALREADY_INITIALIZED
- #define OCOTP_E_BUSY

- #define OCOTP_E_TIMEOUT

  Runtime error codes (passed to DET).
- #define OCOTP_E_WRITE_FAILED

  Transient Faults codes (passed to DET).
- #define OCOTP_E_READ_FAILED
- #define OCOTP_INIT_ID

  All service IDs (passed to DET).
- #define OCOTP_DEINIT_ID
- #define OCOTP_READ_EFUSE_ID
- #define OCOTP_WRITE_MIRROR_REGISTER_ID
- #define OCOTP_READ_MIRROR_REGISTER_ID
- #define OCOTP_GET_STATUS_ID

Types Reference

- typedef uint32 Ocotp_AddressType

  Ocotp address type.
- typedef uint8 Ocotp_ChannelType

  Ocotp channel type.

Enum Reference

- enum Ocotp_HardwareStatusType

  Status of Ocotp hardware.

Function Reference

- void Ocotp_Init (const Ocotp_ConfigType *const ConfigPtr)

  The function initializes Ocotp module.
- void Ocotp_Deinit (void)

  The function de-initializes Ocotp module.
- Std_ReturnType Ocotp_ReadEFuse (Ocotp_ChannelType eFuseChannel, uint32 *pData)

  The function reads data from efuse word.
- Std_ReturnType Ocotp_ReadMirrorRegister (Ocotp_ChannelType eFuseChannel, uint32 *pData)

  The function reads data from the mirror.
- Std_ReturnType Ocotp_WriteMirrorRegister (Ocotp_ChannelType eFuseChannel, uint32 data)

  The function writes data to Mirror register.
- Std_ReturnType Ocotp_GetStatus (uint8 HwIndex, Ocotp_StatusType *pStatus)

  The function gets status of Ocotp module.

6.1.2 Data Structure Documentation

6.1.2.1 struct Ocotp_ChannelConfigType

Ocotp channel configuration type.

A structure which is used to contain the parameters of channel used. Such as address of eFuse. Index of shadow register.

Definition at line 149 of file Ocotp_Types.h.

Data Fields

Type	Name	Description
uint32	Ocotp_MirrorAddress	Address of Mirror Register
uint8	Ocotp_ChannelNumber	Channel number
Ocotp_InstanceType	Ocotp_Instance	Instance Index

6.1.2.2 struct Ocotp_ConfigType

Ocotp configuration type.

A structure which is used to contain the hardware configuration and the configuration of channels used.

Definition at line 162 of file Ocotp_Types.h.

Data Fields

Type	Name	Description
const Mem_Otp_Ip_ConfigType *	pHwConfig	User configuration structure
const Ocotp_ChannelConfigType(*	pChanelConfig)[]	Ocotp channel configuration type

6.1.2.3 struct Ocotp_StatusType

Ocotp status type.

A structure which is used to contain the status of hardware (Busy or idle or errors occurred) and status of repair error flag(If FBXC exists).

Definition at line 174 of file Ocotp_Types.h.

6.1.3 Macro Definition Documentation

6.1.3.1 OCOTP_MODULE_ID

```
#define OCOTP_MODULE_ID
```

ID of module and instance.

Ocotp module ID

Definition at line 145 of file CDD_Ocotp.h.

6.1.3.2 OCOTP_INSTANCE_ID

```
#define OCOTP_INSTANCE_ID
```

Ocotp instance ID

Definition at line 146 of file CDD_Ocotp.h.

6.1.3.3 OCOTP_INVALID_INDEX

```
#define OCOTP_INVALID_INDEX
```

Invalid value of index and address.

Ocotp invalid index ID

Definition at line 151 of file CDD_Ocotp.h.

6.1.3.4 OCOTP_INVALID_ADDRESS

```
#define OCOTP_INVALID_ADDRESS
```

Ocotp invalid address ID

Definition at line 152 of file CDD_Ocotp.h.

6.1.3.5 OCOTP_E_PARAM_CONFIG

```
#define OCOTP_E_PARAM_CONFIG
```

Development error codes (passed to DET).

API service called with wrong parameter

Definition at line 158 of file CDD_Ocotp.h.

6.1.3.6 OCOTP_E_PARAM_LENGTH

```
#define OCOTP_E_PARAM_LENGTH
```

API service called with wrong parameter

Definition at line 159 of file CDD_Ocotp.h.

6.1.3.7 OCOTP_E_PARAM_POINTER

```
#define OCOTP_E_PARAM_POINTER
```

API service called with invalid pointer

Definition at line 160 of file CDD_Ocotp.h.

6.1.3.8 OCOTP_E_PARAM_CHANNEL

```
#define OCOTP_E_PARAM_CHANNEL
```

API service used with an invalid channel identifier or channel was not configured for the functionality of the calling API

Definition at line 161 of file CDD_Ocotp.h.

6.1.3.9 OCOTP_E_PARAM_INSTANCE

```
#define OCOTP_E_PARAM_INSTANCE
```

API service used with an invalid hardware unit identifier or hardware unit was not configured for the functionality of the calling API

Definition at line 162 of file CDD_Ocotp.h.

6.1.3.10 OCOTP_E_PARAM_MODE

```
#define OCOTP_E_PARAM_MODE
```

API service called with wrong parameter

Definition at line 163 of file CDD_Ocotp.h.

6.1.3.11 OCOTP_E_UNINIT

```
#define OCOTP_E_UNINIT
```

API service called without module initialization

Definition at line 164 of file CDD_Ocotp.h.

6.1.3.12 OCOTP_E_ALREADY_INITIALIZED

```
#define OCOTP_E_ALREADY_INITIALIZED
```

Ocotp_init service called module initialization

Definition at line 165 of file CDD_Ocotp.h.

6.1.3.13 OCOTP_E_BUSY

```
#define OCOTP_E_BUSY
```

API service called while driver still busy

Definition at line 166 of file CDD_Ocotp.h.

6.1.3.14 OCOTP_E_TIMEOUT

```
#define OCOTP_E_TIMEOUT
```

Runtime error codes (passed to DET).

Timeout exceeded

Definition at line 171 of file CDD_Ocotp.h.

6.1.3.15 OCOTP_E_WRITE_FAILED

```
#define OCOTP_E_WRITE_FAILED
```

Transient Faults codes (passed to DET).

Ocotp write failed (HW)

Definition at line 177 of file CDD_Ocotp.h.

6.1.3.16 OCOTP_E_READ_FAILED

```
#define OCOTP_E_READ_FAILED
```

Ocotp read failed (HW)

Definition at line 178 of file CDD_Ocotp.h.

6.1.3.17 OCOTP_INIT_ID

```
#define OCOTP_INIT_ID
```

All service IDs (passed to DET).

Service ID of Ocotp_Init function

Definition at line 183 of file CDD_Ocotp.h.

6.1.3.18 OCOTP_DEINIT_ID

```
#define OCOTP_DEINIT_ID
```

Service ID of Ocotp_Deinit function

Definition at line 184 of file CDD_Ocotp.h.

6.1.3.19 OCOTP_READ_EFUSE_ID

```
#define OCOTP_READ_EFUSE_ID
```

Service ID of Ocotp_ReadEFuse function

Definition at line 185 of file CDD_Ocotp.h.

6.1.3.20 OCOTP_WRITE_MIRROR_REGISTER_ID

```
#define OCOTP_WRITE_MIRROR_REGISTER_ID
```

Service ID of Ocotp_WriteMirrorRegister function

Definition at line 186 of file CDD_Ocotp.h.

6.1.3.21 OCOTP_READ_MIRROR_REGISTER_ID

```
#define OCOTP_READ_MIRROR_REGISTER_ID
```

Service ID of Ocotp_ReadMirrorRegister function

Definition at line 187 of file CDD_Ocotp.h.

6.1.3.22 OCOTP_GET_STATUS_ID

```
#define OCOTP_GET_STATUS_ID
```

Service ID of Ocotp_GetStatus function

Definition at line 188 of file CDD_Ocotp.h.

6.1.4 Types Reference

6.1.4.1 Ocotp_AddressType

```
typedef uint32 Ocotp_AddressType
```

Ocotp address type.

Adrress of a register.

Definition at line 134 of file Ocotp_Types.h.

6.1.4.2 Ocotp_ChannelType

```
typedef uint8 Ocotp_ChannelType
```

Ocotp channel type.

An integer which is used to describe the order of channel configured.

Definition at line 141 of file Ocotp_Types.h.

6.1.5 Enum Reference

6.1.5.1 Ocotp_HardwareStatusType

```
enum Ocotp_HardwareStatusType
```

Status of Ocotp hardware.

Enumerator

OCOTP_HARDWARE_BUSY	Status of hardware is busy
OCOTP_HARDWARE_ERROR	Status of hardware is error
OCOTP_HARDWARE_IDLE	Status of hardware is idle

Definition at line 121 of file Ocotp_Types.h.

6.1.6 Function Reference

6.1.6.1 Ocotp_Init()

```
void Ocotp_Init (
            const Ocotp_ConfigType *const ConfigPtr )
```

The function initializes Ocotp module.

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

Parameters

in	*ConfigPtr*	Pointer to Ocotp driver configuration set.

Precondition

Ocotp_pConfigPtr must be NULL_PTR.

6.1.6.2 Ocotp_Deinit()

```
void Ocotp_Deinit (
            void  )
```

The function de-initializes Ocotp module.

The function sets the Ocotp module's status to un-initialized.

Parameters

in	*none.*	

Precondition

Ocotp_pConfigPtr must not be NULL_PTR.

6.1.6.3 Ocotp_ReadEFuse()

```
Std_ReturnType Ocotp_ReadEFuse (
            Ocotp_ChannelType eFuseChannel,
            uint32 * pData )
```

The function reads data from efuse word.

The function reads the data from the eFuse which was configured by configuration tool.

Parameters

in	*eFuseChannel*	Channel of eFuse such as 0, 1, 2 which was configured by user.
in	*data*	pointer points to the data read from eFuse word.

Returns

Std_ReturnType

Return values

E_OK	Read operation is successful.
E_NOT_OK	Read operation failed.

Precondition

Ocotp_pConfigPtr must not be NULL_PTR.

6.1.6.4 Ocotp_ReadMirrorRegister()

```
Std_ReturnType Ocotp_ReadMirrorRegister (
            Ocotp_ChannelType eFuseChannel,
            uint32 * pData )
```

The function reads data from the mirror.

The function reads the data from the mirror which was configured by configuration tool.

Parameters

in	*eFuseChannel*	Channel of eFuse such as 0, 1, 2 which was configured by user.
in	*data*	pointer points to the data read from mirror word.

Returns

Std_ReturnType

Return values

E_OK	Read operation is successful.
E_NOT_OK	Read operation failed.

Precondition

Ocotp_pConfigPtr must not be NULL_PTR.

6.1.6.5 Ocotp_WriteMirrorRegister()

```
Std_ReturnType Ocotp_WriteMirrorRegister (
            Ocotp_ChannelType eFuseChannel,
            uint32 data )
```

The function writes data to Mirror register.

The function writes the data to the Mirror register which was configured in eFuse channel by configuration tool.

Parameters

in	*eFuseChannel*	Channel of eFuse such as 0, 1, 2 which was configured by user.
in	*data*	data to be written.

Returns

Std_ReturnType

Return values

E_OK	write operation is successful.
E_NOT_OK	write operation failed.

Precondition

Ocotp_pConfigPtr must not be NULL_PTR.

6.1.6.6 Ocotp_GetStatus()

```
Std_ReturnType Ocotp_GetStatus (
            uint8 HwIndex,
            Ocotp_StatusType * pStatus )
```

The function gets status of Ocotp module.

The function gets status of Ocotp module Secure mode is enabled or not. Error occurred or not and Status of hardware.

Parameters

in	*HwIndex*	Index of hardware configured.
in	*pStatus*	Pointer points to status structure.

Returns

Std_ReturnType

Precondition

Ocotp_pConfigPtr must not be NULL_PTR.

6.2 MEM_OTP_IP

6.2.1 Detailed Description

Data Structures

- struct Mem_Otp_Ip_ConfigType
  User configuration structure. More...

Types Reference

- typedef uint8 Ocotp_InstanceType
  Ocotp instance type.

Enum Reference

- enum Mem_Otp_Ip_StatusType
  Ocotp Ip status type.

Function Reference

- void Mem_Otp_Ip_Init (const Mem_Otp_Ip_ConfigType *pConfig)
  The function initializes Mem_Otp_Ip module.
- void Mem_Otp_Ip_DeInit (uint32 instance)
  The function de-initializes Mem_Otp_Ip module.
- Mem_Otp_Ip_StatusType Mem_Otp_Ip_WriteMirrorRegister (uint32 instance, uint32 address, uint32 data)
  The function write data in a specified mirror register.
- Mem_Otp_Ip_StatusType Mem_Otp_Ip_ReadMirrorRegister (uint32 instance, uint32 address, uint32 *data)
  The function reads data in a specified mirror register.
- Mem_Otp_Ip_StatusType Mem_Otp_Ip_ReadOTPMem (uint32 instance, uint32 address, uint32 *data)
  The function reads a single byte of data from OTP memory.
- Mem_Otp_Ip_StatusType Mem_Otp_Ip_GetBusyState (uint32 instance)
  The function checks busy and error state of hardware.

6.2.2 Data Structure Documentation

6.2.2.1 struct Mem_Otp_Ip_ConfigType

User configuration structure.

Structure contains the configuration parameters which will be used to initialize Ocotp module.

Definition at line 106 of file Mem_Otp_Ip_Types.h.

Data Fields

Type	Name	Description
Ocotp_InstanceType	ocotpInstance	Ocotp instance

6.2.3 Types Reference

6.2.3.1 Ocotp_InstanceType

```
typedef uint8 Ocotp_InstanceType
```

Ocotp instance type.

An integer which describe the order of the instance configured.

Definition at line 100 of file Mem_Otp_Ip_Types.h.

6.2.4 Enum Reference

6.2.4.1 Mem_Otp_Ip_StatusType

```
enum Mem_Otp_Ip_StatusType
```

Ocotp Ip status type.

Enumerator

STATUS_MEM_OTP_IP_ERROR	Boot Error
STATUS_MEM_OTP_IP_BUSY	Status busy
STATUS_MEM_OTP_IP_SUCCESS	Status success
STATUS_MEM_OTP_IP_INVALID_INPUT_ADDRESS	invalid input parameter
STATUS_MEM_OTP_IP_ERROR_TIMEOUT	timeout error

Definition at line 84 of file Mem_Otp_Ip_Types.h.

6.2.5 Function Reference

6.2.5.1 Mem_Otp_Ip_Init()

```
void Mem_Otp_Ip_Init (
            const Mem_Otp_Ip_ConfigType * pConfig )
```

The function initializes Mem_Otp_Ip module.

The function clear all errors flag and initializes Mem_Otp_Ip module.

Parameters

in	*pConfig*	pointer points to configuration structure.

Precondition

Module has been initialized.

6.2.5.2 Mem_Otp_Ip_DeInit()

```
void Mem_Otp_Ip_DeInit (
            uint32 instance )
```

The function de-initializes Mem_Otp_Ip module.

The function de-initializes Mem_Otp_Ip module.

Parameters

in	*instance*	hardware instance.

Precondition

Module has been initialized.

6.2.5.3 Mem_Otp_Ip_WriteMirrorRegister()

```
Mem_Otp_Ip_StatusType Mem_Otp_Ip_WriteMirrorRegister (
            uint32 instance,
            uint32 address,
            uint32 data )
```

The function write data in a specified mirror register.

The function write data in a specified mirror register with an OTPC command or direct register access.

Parameters

in	*instance*	hardware instance.
in	*address*	address of mirror.
in	*data*	data to be written.

Returns

Mem_Otp_Ip_StatusType

Return values

STATUS_MEM_OTP_IP_SUCCESS	if write operation is successful.
STATUS_MEM_OTP_IP_ERROR	if error flag is set.
STATUS_MEM_OTP_IP_BUSY	hardware is busy.

Precondition

Module has been initialized.

6.2.5.4 Mem_Otp_Ip_ReadMirrorRegister()

```
Mem_Otp_Ip_StatusType Mem_Otp_Ip_ReadMirrorRegister (
            uint32 instance,
            uint32 address,
            uint32 * data )
```

The function reads data in a specified mirror register.

The function reads data in a specified mirror register with an OTPC command or direct register access.

Parameters

in	*instance*	hardware instance.
in	*address*	address of mirror.
out	*data*	pointer points to data result.

Returns

Mem_Otp_Ip_StatusType

Return values

STATUS_MEM_OTP_IP_SUCCESS	if read operation is successful.
STATUS_MEM_OTP_IP_ERROR	if error flag is set.
STATUS_MEM_OTP_IP_BUSY	hardware is busy.

Precondition

Module has been initialized.

6.2.5.5 Mem_Otp_Ip_ReadOTPMem()

```
Mem_Otp_Ip_StatusType Mem_Otp_Ip_ReadOTPMem (
            uint32 instance,
            uint32 address,
            uint32 * data )
```

The function reads a single byte of data from OTP memory.

The function reads a single byte of data from OTP memory.

Parameters

in	*instance*	hardware instance.
in	*address*	address of OTP memory.
out	*data*	pointer points to data result.

Returns

Mem_Otp_Ip_StatusType

Return values

STATUS_MEM_OTP_IP_SUCCESS	if read operation is successful.
STATUS_MEM_OTP_IP_ERROR	if error flag is set.
STATUS_MEM_OTP_IP_BUSY	hardware is busy.

Precondition

Module has been initialized.

6.2.5.6 Mem_Otp_Ip_GetBusyState()

```
Mem_Otp_Ip_StatusType Mem_Otp_Ip_GetBusyState (
            uint32 instance )
```

The function checks busy and error state of hardware.

The function checks busy and error state of hardware.

Parameters

in	*instance*	hardware instance.

Returns

Mem_Otp_Ip_StatusType

Return values

STATUS_MEM_OTP_IP_SUCCESS	if hardware is ready.
STATUS_MEM_OTP_IP_ERROR	if error flag is set.
STATUS_MEM_OTP_IP_BUSY	hardware is busy.

Precondition

Module has been initialized.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support