Document Title	Acceptance Test Specification of Memory Stack
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	669
Document Classification	Auxiliary

Document Status	Final
Part of AUTOSAR Standard	Acceptance Tests for Classic Platform
Part of Standard Release	1.2.0

Document Change History

Date	**Release**	**Changed by**	**Change Description**
2016-12-15	1.2.0	AUTOSAR Release Management	• Checked for applicability for Classic Platform Release 4.2.2
2015-10-31	1.1.0	AUTOSAR Release Management	• Formalize point of control and observation for all test cases • Some test steps were split to ensure that atomic test steps have a single PCO for their execution and their pass criteria • Checked and adapted to Classic Platform Release 4.2.1 • Formal changes
2014-07-30	1.0.0	AUTOSAR Release Management	• Initial release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
API	Application Programming Interface
AT	Acceptance Test
BSW	Basic Software
CAN	Controller Area Network
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
ECU	Electronic Control Unit
LT	Lower Tester
NM	Network Management
NvM	Nonvolatile RAM Manager
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PIM	Per-Instance Memory
RfC	Request for Change
RAM	Random Access Memory
ROM	Read-only Memory
Rx	Reception
SUT	System Under Test
SWC	Software Component
TCP	Test Coordination Procedures
Tx	Transmission
UT	Upper Tester

2 Scope

The following test cases are used to verify the correct behavior of all the memory stack features.

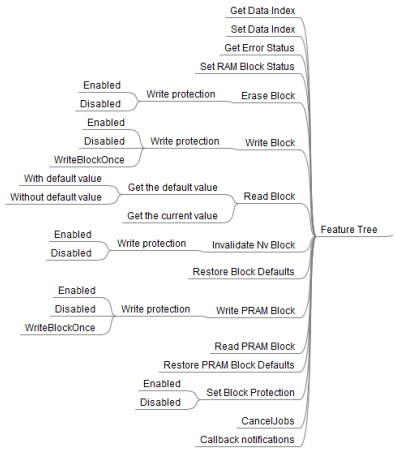
Each test case documents for which releases of the AUTOSAR software specification it can be used:

- When test cases are known to be applicable for a release, this is mentioned in the “AUTOSAR Releases” field of the test case specifications.
  You can find a summary of the applicability of all test cases to the software specification releases in the “AUTOSAR_TR_ATSReleaseApplicability” document.
- When test cases are known to require adaptations (in their configuration requirements or test sequences), this is mentioned in the “Needed Adaptation to other Releases” field of the test case specifications.

3 RS_BRF_01416 – NvM services

3.1 General Test Objective and Approach

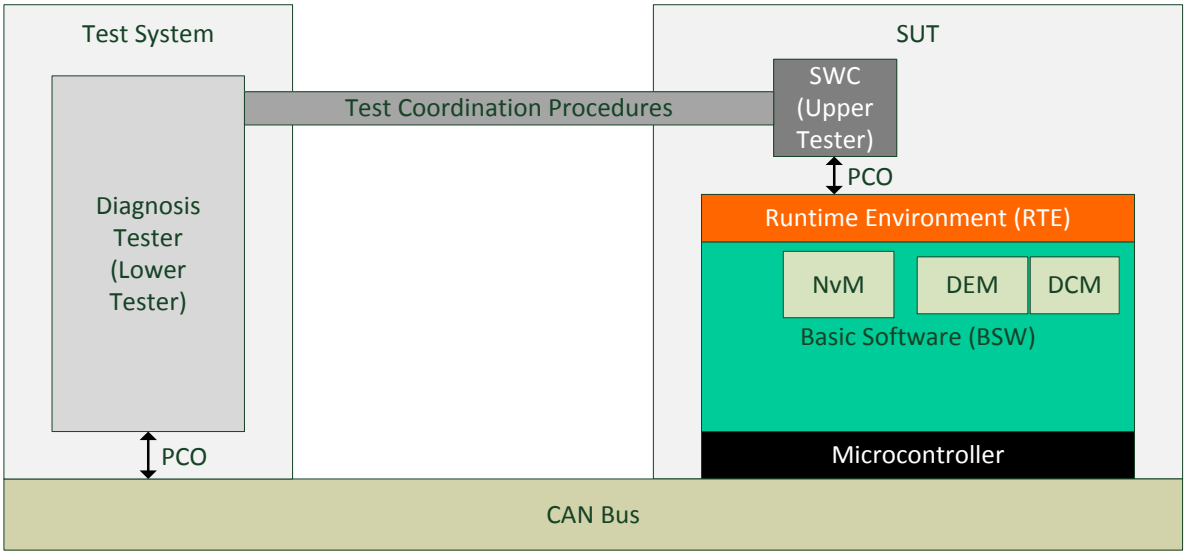
This test specification intends to cover the memory services features.
This test case document has been established to cover the following features:



This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

3.1.1 Test System

3.1.1.1 Overview on Architecture



The test system architecture consists of testing the read and write in different memory block type.

3.1.1.2 Specific Requirements

Not Applicable

3.1.1.3 Test Coordination Requirements

Not Applicable

3.1.2 Test Configuration

This section describes sets of requirements on configuration. These sets are later referenced by test cases. No configuration files are provided. They need to be developed when the test suite is implemented.

3.1.2.1 Required ECU Extract of System Description Files

Execution of the test cases requires only one configuration set to be created.

3.1.2.2 Required ECU Configuration Description Files

The ECU Extract file will describe one Tester SWC which uses client / Server connections to NvM Service component.

Each block is accessible through a dedicated port interface with port api option that allows calling to NvM APIs with a specific BlockId.

3.1.2.2.1 Test Case AT-120: Read/Write/Erase Native Block

For this test case, the configuration should be the following:

Per Instance Memory : NvBlock1_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block1 (Native Block)		
Assigned Datas	Used Pim: NvBlock1_Data	
Assigned Port	NativeBlock1_Port	(Port API option : BlockId = NVBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	For this test, we need that Native block not protected by CRC
reliability	noProtection	

Per Instance Memory : NvBlock2_Data		
NvBlock2_Data	Uint16	
NV Block Needs : Block1 (Native Block)		
Assigned Datas	Used Pim: NvBlock2_Data	
Assigned Port	NativeBlock2_Port	(Port API option : BlockId = NVBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	For this test, we need that Native block not protected by CRC
reliability	noProtection	

3.1.2.2.2 Test Case AT-197: Read/Write/Erase Redundant Block

For this test case, the configuration should be the following:

Per Instance Memory : RedundantBlock1_Data		
RedundantBlock1_Data	Uint16	
NV Block Needs : redundant Block1 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock1_Data	
Assigned Port	RedundantBlock1_Port	(Port API option : BlockId = REDUNDANTBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	As defined in SWC Template mapping rules, the combination of nDataSet=0 & reliability = errorCorrection implies usage of Redundant Block
reliability	errorCorrection	

Per Instance Memory : RedundantBlock2_Data		
RedundantBlock2_Data	Uint16	
NV Block Needs : Block1 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock2_Data	
Assigned Port	RedundantBlock2_Port	(Port API option : BlockId = NVBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	As defined in SWC Template mapping rules, the combination of nDataSet=0 & reliability = errorCorrection implies usage of Redundant Block
reliability	errorCorrection	

3.1.2.2.3 Test Case AT-198: Write Protection on Native Block

For this test case, the configuration should be the following:

Per Instance Memory : WPNvBlock1_Data		
WPNvBlock1_Data	Uint16	
NV Block Needs : WPNvBlock1 (Write Protected Block)		
Assigned Datas	Used Pim: WPNvBlock1_Data	
Assigned Port	WPNvBlock1_Port	(Port API option : BlockId = WPNATIVEBLOCK1)
Readonly	True	**Write Protected**
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	
calcRamBlockCrc	False	
nDataSet	0	
reliability	noProtection	

Per Instance Memory : WPNvBlock2_Data		
WPNvBlock2_Data	Uint16	
NV Block Needs : WPNvBlock2 (Write only once Block)		
Assigned Datas	Used Pim: RedundantBlock2_Data	
Assigned Port	RedundantBlock2_Port	(Port API option : BlockId = WPNATIVEBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	True	Write Only Once block
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	
reliability	noProtection	

3.1.2.2.4 Test Case AT-199: Read/Write/Erase DataSet Block

For this test case, the configuration should be the following:

Per Instance Memory : DataSetBlock1_Data		
DataSetBlock1_Data	Uint16	
NV Block Needs : redundant Block1 (Native Block)		
Assigned Datas	Used Pim: DataSetBlock1_Data	
Assigned Port	DataSetBlock1_Port	(Port API option : BlockId = DATASETBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	
calcRamBlockCrc	False	
nDataSet	2	
reliability	noProtection	

3.1.2.2.5 Test Case AT-202: Job Notification on Native Block

For this test case, the configuration should be the following:

Per Instance Memory : NvBlock1_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block1 (Native Block)		
Assigned Datas	Used Pim: NvBlock1_Data	
Assigned Port	NativeBlock1_Port	(Port API option : BlockId = NVBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	True	
nDataSet	0	
Reliability	errorDetection	

3.1.2.2.6 Test Case AT-203: Explicit Restore default values on Native Block

For this test case, the configuration should be the following:

Per Instance Memory : NvBlock1_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block1 (Native Block)		
Assigned Datas	Used Pim: NvBlock1_Data	
Assigned Port	NativeBlock1_Port	(Port API option : BlockId = NVBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorDetection	

Per Instance Memory : NvBlock2_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block2 (Native Block)		
Assigned Datas	Used Pim: NvBlock2_Data	
Assigned Port	NativeBlock2_Port	(Port API option : BlockId = NVBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorDetection	

Per Instance Memory : NvBlock3_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block3 (Native Block)		
Assigned Datas	Used Pim: NvBlock3_Data	
Assigned Port	NativeBlock3_Port	(Port API option : BlockId = NVBLOCK3)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorDetection	

3.1.2.2.7 Test Case AT-204: Implicit Restore default values on Native Block

For this test case, the configuration should be the following:

Per Instance Memory : NvBlock1_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block1 (Native Block)		
Assigned Datas	Used Pim: NvBlock1_Data	
Assigned Port	NativeBlock1_Port	(Port API option : BlockId = NVBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorDetection	

Per Instance Memory : NvBlock2_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block2 (Native Block)		
Assigned Datas	Used Pim: NvBlock2_Data	
Assigned Port	NativeBlock2_Port	(Port API option : BlockId = NVBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorDetection	

Per Instance Memory : NvBlock3_Data		
NvBlock1_Data	Uint16	
NV Block Needs : Block3 (Native Block)		
Assigned Datas	Used Pim: NvBlock3_Data	
Assigned Port	NativeBlock3_Port	(Port API option : BlockId = NVBLOCK3)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorDetection	

3.1.2.2.8 Test Case AT-250: Write Protection on Redundant Block

For this test case, the configuration should be the following:

Per Instance Memory : WPRedundantBlock1_Data		
WPNvBlock1_Data	Uint16	
NV Block Needs : WPRedundantBlock1 (Write Protected Block)		
Assigned Datas	Used Pim: WPRedundantBlock1_Data	
Assigned Port	WPRedundantBlock1_Port	(Port API option : BlockId = WPREDUNDANTBLOCK1)
Readonly	True	**Write Protected**
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	
calcRamBlockCrc	False	
nDataSet	0	
reliability	errorCorrection	

Per Instance Memory : WPRedundantBlock2_Data		
WPRedundantBlock2_Data	Uint16	
NV Block Needs : WPRedundantBlock2 (Write only once Block)		
Assigned Datas	Used Pim: WPRedundantBlock2_Data	
Assigned Port	WPRedundantBlock2_Port	(Port API option : BlockId = WPREDUNDANTBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	True	Write Only Once block
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	
reliability	errorCorrection	

3.1.2.2.9 Test Case AT-251: Write Protection on DataSet Block

For this test case, the configuration should be the following:

Per Instance Memory : WPDataSetBlock1_Data		
WPDataSetBlock1_Data	Uint16	
NV Block Needs : WPDataSetBlock1 (Write Protected Block)		
Assigned Datas	Used Pim: WPDataSetBlock1_Data	
Assigned Port	WPDataSetBlock1_Port	(Port API option : BlockId = WPDATASETBLOCK1)
Readonly	True	**Write Protected**
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	
calcRamBlockCrc	False	
nDataSet	2	
reliability	errorDetection	

Per Instance Memory : WPDataSetBlock2_Data		
WPDataSetBlock2_Data	Uint16	
NV Block Needs : WPDataSetBlock2 (Write only once Block)		
Assigned Datas	Used Pim: WPDataSetBlock2_Data	
Assigned Port	WPDataSetBlock2_Port	(Port API option : BlockId = WPDATASETBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	True	Write Only Once block
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	
calcRamBlockCrc	False	
nDataSet	2	
reliability	errorDetection	

3.1.2.2.10 Test Case AT-252: Job Notification on DataSet Block

For this test case, the configuration should be the following:

Per Instance Memory : DataSetBlock1_Data		
DataSetBlock1_Data	Uint16	
NV Block Needs : DataSetBlock1 (DataSetBlock)		
Assigned Datas	Used Pim: DataSetBlock1_Data	
Assigned Port	DataSetBlock1_Port	(Port API option : BlockId = DATASETBLOCK1)
Readonly	False	
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	
calcRamBlockCrc	False	
nDataSet	2	
reliability	errorDetection	

3.1.2.2.11 Test Case AT-253: Job Notification on Redundant Block

For this test case, the configuration should be the following:

Per Instance Memory : RedundantBlock1_Data		
RedundantBlock1_Data	Uint16	
NV Block Needs : redundant Block1 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock1_Data	
Assigned Port	RedundantBlock1_Port	(Port API option : BlockId = REDUNDANTBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	False	
nDataSet	0	As defined in SWC Template mapping rules, the combination of nDataSet=0 & reliability = errorCorrection implies usage of Redundant Block
reliability	errorCorrection	

3.1.2.2.12 Test Case AT-254: Explicit Restore default values on Redundant Block

For this test case, the configuration should be the following:

Per Instance Memory : RedundantBlock1_Data		
RedundantBlock1_Data	Uint16	
NV Block Needs : Block1 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock1_Data	
Assigned Port	RedundantBlock1_Port	(Port API option : BlockId = REDUNDANTBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorCorrection	

Per Instance Memory : RedundantBlock2_Data		
RedundantBlock2_Data	Uint16	
NV Block Needs : Block2 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock2_Data	
Assigned Port	RedundantBlock2_Port	(Port API option : BlockId = REDUNDANTBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorCorrection	

Per Instance Memory : RedundantBlock3_Data		
RedundantBlock3_Data	Uint16	
NV Block Needs : Block3(Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock3_Data	
Assigned Port	RedundantBlock3_Port	(Port API option : BlockId =

		REDUNDANTBLOCK3)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorCorrection	

3.1.2.2.13 Test Case AT-255: Explicit Restore default values on DataSet Block

For this test case, the configuration should be the following:

Per Instance Memory : DataSetBlock1_Data		
DataSetBlock1_Data	Uint16	
NV Block Needs : DataSetBlock1 (DataSet Block)		
Assigned Datas	Used Pim: DataSetBlock1_Data	
Assigned Port	DataSetBlock1_Port	(Port API option : BlockId = DATASETBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	Default value = Index_1= 0x1234 Index_2 = 0x1234
calcRamBlockCrc	True	
nDataSet	2	
reliability	errorDetection	
reliability	errorDetection	

Per Instance Memory : DataSetBlock2_Data		
DataSetBlock2_Data	Uint16	
NV Block Needs : DataSetBlock2 (DataSet Block)		
Assigned Datas	Used Pim: DataSetBlock2_Data	
Assigned Port	DataSetBlock2_Port	(Port API option : BlockId = DATASETBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	Default value = Index_1= 0x1234, Index_2 = 0x1234
calcRamBlockCrc	True	
nDataSet	2	
reliability	errorDetection	

3.1.2.2.14 Test Case AT-256: Implicit Restore default values on Redundant Block

For this test case, the configuration should be the following:

Per Instance Memory : RedundantBlock1_Data		
RedundantBlock1_Data	Uint16	
NV Block Needs : Block1 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock1_Data	
Assigned Port	RedundantBlock1_Port	(Port API option : BlockId = REDUNDANTBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorCorrection	

Per Instance Memory : RedundantBlock2_Data		
RedundantBlock2_Data	Uint16	
NV Block Needs : Block2 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock2_Data	
Assigned Port	RedundantBlock2_Port	(Port API option : BlockId = REDUNDANTBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	1	Default value = 0x1234
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorCorrection	

Per Instance Memory : RedundantBlock3_Data		
RedundantBlock3_Data	Uint16	
NV Block Needs : Block3 (Redundant Block)		
Assigned Datas	Used Pim: RedundantBlock3_Data	
Assigned Port	RedundantBlock3_Port	(Port API option : BlockId = REDUNDANTBLOCK3)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	0	
calcRamBlockCrc	True	
nDataSet	0	
reliability	errorCorrection	

3.1.2.2.15 Test Case AT-257: Implicit Restore default values on DataSet Block

For this test case, the configuration should be the following:

Per Instance Memory : DataSetBlock1_Data		
DataSetBlock1_Data	Uint16	
NV Block Needs : DataSetBlock1 (DataSet Block)		
Assigned Datas	Used Pim: DataSetBlock1_Data	
Assigned Port	DataSetBlock1_Port	(Port API option : BlockId = DATASETBLOCK1)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	Default value = Index_1= 0x1234 Index_2 = 0x1234
calcRamBlockCrc	True	
nDataSet	2	
reliability	errorDetection	
reliability	errorDetection	

Per Instance Memory : DataSetBlock2_Data		
DataSetBlock2_Data	Uint16	
NV Block Needs : DataSetBlock2 (DataSet Block)		
Assigned Datas	Used Pim: DataSetBlock2_Data	
Assigned Port	DataSetBlock2_Port	(Port API option : BlockId = DATASETBLOCK2)
Readonly	False	Not Write Protected
ResistantToChangedSw	False	
WriteOnlyOnce	False	
RestoreAtStart	False	
StoreAtShutdown	False	
ramBlockStatusControl	nvRamManager	
nRomBlocks	2	Default value = Index_1= 0x1234 Index_2 = 0x1234
calcRamBlockCrc	True	
nDataSet	2	
reliability	errorDetection	

3.1.2.3 Required Software Component Description Files

No specific configuration requirements for Software Components.

3.1.2.4 Mandatory vs. Customizable Parts

In the configuration set, the only mandatory definition is to have at least 2 blocks of each NvM type (Native, Redundant, DataSet)

3.1.3 Test Case Design

The Tests cases rely on the following assumptions:

- ➔ The Selected ECU for SUT can write / erase Non Volatile Blocks of device in less than 1 second
- ➔ The Test suite uses DCM/DEM BSW components services to get default codes generated by NvM Service
- ➔ Used ROM Blocks are initialized with the Default values 0x1234
- ➔ All services interfaces are defined as synchronous even for NvM_NotifyJobFinished which acts as server on SWC user side.

3.2 Re-usable Test Steps

Not Applicable

3.3 Test Cases

3.3.1 [ATS_MEM_00120] Read, write and erase native block

Test Objective	Read, write and erase native block		
ID	ATS_MEM_00120	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00035		
Trace to SWS Item	NVRAMManager: SWS_NvM_00000 NVRAMManager: SWS_NvM_00698 NVRAMManager: SWS_NvM_00699 NVRAMManager: SWS_NvM_00734		
Requirements / Reference to Test Environment	UC06.05.11, UC06.05.13		
Configuration Parameters	SWC uses 2 Native Blocks configured to have immediate priority (required for successful erase operation). Details are given in ATS Specification chapter 3.1.2.2.1 AT-120. For readability, the NvBlockx will be replaced in this Test case by Blockx. Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the management of NATIVE block by - Writing different data in two Native NvBlock (and switch-off/ switch-on the power) - Read data in each NvBlock - Erase and read data on each NvBlock		
Needed Adaptation to other Releases	None		
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data1 value		
Step 2	[SWC] Call NvM_WriteBlock() Service to transfer data in Block_1		[SWC] NvM_WriteBlock should return E_OK
Step 3	[CP] WAIT 1s		
Step 4	[SWC] Execute NvM_GetErrorStatus() on Block_1		[SWC] GetErrorStatus returned value should be E_OK RequestResult parameter should be equal to NVM_REQ_OK
Step 5	[SWC] Fill Pim Data2 with the Data_2 value		
Step 6	[SWC] Call NvM_WriteBlock() service for Block_2 with Data_2 value		[SWC] NvM_WriteBlock() should return E_OK

Step 7	[CP] WAIT 1s	
Step 8	[SWC] Call NvM_GetErrorStatus() service on Block_2	[SWC] Returned value should be E_OK RequestResult parameter should be equal to NVM_REQ_OK
Step 9	[LT<Power Supply>] Power off ECU	
Step 10	[CP] Wait n seconds (n depends on time needed for platform to shutdown)	
Step 11	[LT<Power Supply>] Power on ECU	
Step 12	[SWC] Write 0xFFFF value in Pim Data1 & Data2	
Step 13	[SWC] Call NvM_ReadBlock() service on Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 14	[CP] WAIT 1s	
Step 15	[SWC] Call NvM_GetErrorStatus() service on Block_1	[SWC] Returned value should be E_OK RequestResult parameter should be equal to NVM_REQ_OK
Step 16	[SWC] Check the Pim Data1	[SWC] Data Pim should be equal to Data_1 value
Step 17	[SWC] Call NvM_ReadBlock() service on Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 18	[CP] Wait 1s	
Step 19	[SWC] Call NvM_GetErrorStatus() service on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 20	[SWC] Check Pim Data2 value	[SWC] Pim Data2 should be equal to Data_2 value
Step 21	[SWC] Call NvM_EraseNvBlock() service on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 22	[CP] Wait 1s	
Step 23	[SWC] Call NvM_GetErrorStatus() service on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 24	[SWC] Call NvM ReadBlock() Service on Block_1	[SWC] Return value should be E_OK
Step 25	[CP] Wait 1s	
Step 26	[SWC]	[SWC]

	Call NvM GetErrorStatus() service on Block_1	NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR Value in Block1_Data should be equal to previous value (0x55AA)
Step 27	[SWC] Call NvM_EraseNvBlock() on Block_2	[SWC] NvM_EraseNvBlock() should return E_OK
Step 28	[CP] Wait 1s	
Step 29	[SWC] Call NvM_GetErrorStatus() service on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 30	[SWC] Call NvM_ReadBlock() service on Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 31	[CP] Wait 1s	
Step 32	[SWC] Call NvM_GetErrorStatus() service on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR Value in Block2_Data should be equal to previous value (0xAA55)
Post-conditions	None	

3.3.2 [ATS_MEM_00197] Read, write and erase redundant block

Test Objective	Read, write and erase redundant block		
ID	ATS_MEM_00197	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00035		
Trace to SWS Item	NVRAMManager: SWS_NvM_00001 NVRAMManager: SWS_NvM_00698 NVRAMManager: SWS_NvM_00699 NVRAMManager: SWS_NvM_00734		
Requirements /	UC06.05.21, UC06.05.23		

Reference to Test Environment	
Configuration Parameters	SWC uses 2 Redundant Blocks configured to have immediate priority (required for successful erase operation). Details are given in ATS Specification chapter 3.1.2.2.2 AT-197. For readability, the RedundantBlockx will be replaced in this Test case by Blockx. Data_1 value =0x55AA Data_2 value =0xAA55
Summary	The aim of this test is to verify the management of REDUNDANT block by - Writing different data in two redundant NvBlocks (and switch-off/ switch-on the power) - Read data in each NvBlock - Erase and read data on each NvBlock This test must be done on NvBlock and Permanent NvBlock.
Needed Adaptation to other Releases	None
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data_1 value	
Step 2	[SWC] Call NvM Service WriteBlock() for Block_1	[SWC] NvM Service call should return E_OK
Step 3	[CP] Wait 2s	
Step 4	[SWC] Call NvM Service GetErrorStatus() on NvBlock_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 5	[SWC] Fill Pim Data2 with Data_2 value	
Step 6	[SWC] Call NvM Service WriteBlock() for Block_2	[SWC] NvM Service call should return E_OK
Step 7	[CP] Wait 2s	
Step 8	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 9	[LT<Power Supply>] Power off ECU	
Step 10	[CP] Wait n seconds (n depends on time needed for platform to shutdown)	
Step 11	[LT<Power Supply>] Power on ECU	
Step 12	[SWC] Write 0xFFFF value in Pim Data1 & Data2	
Step 13	[SWC]	[SWC]

	RUN_Ctrl executes NvM_ReadBlock() on Block_1	NvM_ReadBlock() should return E_OK
Step 14	[CP] Wait 1s	
Step 15	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 16	[SWC] Check PIM Data1	[SWC] Data1 should be equal to Data_1
Step 17	[SWC] Call NvM_ReadBlock() on Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 18	[CP] WAIT 1s	
Step 19	[SWC] Call NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 20	[SWC] Check Pim Data2	[SWC] Pim Data2 should be equal to Data_2
Step 21	[SWC] Call NvM_EraseNvBlock on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 22	[CP] WAIT 2s	
Step 23	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 24	[SWC] Call NvM_EraseNvBlock() on Block_2	[SWC] NvM_EraseNvBlock() should return E_OK
Step 25	[CP] Wait 2s	
Step 26	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 27	[SWC] Execute NvM_ReadBlock() on Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 28	[CP] Wait 1s	
Step 29	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR
Step 30	[SWC]	[SWC]

	Execute NvM_ReadBlock() on Block_2	NvM_ReadBlock() should return E_OK
Step 31	[CP] Wait 1s	
Step 32	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR
Post-conditions	None	

3.3.3 [ATS_MEM_00198] Write protection on Native Block

Test Objective	Write protection on Native Block		
ID	ATS_MEM_00198	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00034		
Trace to SWS Item	NVRAMManager: SWS_NvM_00325 NVRAMManager: SWS_NvM_00578 NVRAMManager: SWS_NvM_00734		
Requirements / Reference to Test Environment	UC06.05.11,		
Configuration Parameters	SWC uses 2 Native Blocks (1 Write protected, 1 WriteOnlyOnce). Details are given in ATS Specification chapter 3.1.2.2.3 AT-198. For better readability, the WPNvBlockx is replaced in this Test case by Blockx. Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the write block protection: • A write request is done to a write-protected block. This is expected to be rejected. • A write request is done to a WriteOnlyOnce block (that has never been written before). This is expected to be accepted and processed correctly. • A second write request is done to a WriteOnlyOnce block. This is expected to be rejected. • The write protection is deactivated for the write-protected block. • A write request is done to the formerly write-protected block. This is expected to be accepted and processed correctly.		
Needed Adaptation to other Releases	None		
Pre-conditions	The WriteOnlyOnce block has never been written before.		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with the Data_1 value	
Step 2	[SWC] Execute NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_NOT_OK (as block is write protected)
Step 3	[SWC] Check status of DEM Event NVM_E_WRITE_PROTECTED.	[SWC] Status Bit "TestFailed" of DEM Event NVM_E_WRITE_PROTECTED is set.
Step 4	[SWC] Fill Pim Data2 with Data_1 value	
Step 5	[SWC] Call NvM WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_OK (first write in Block should succeed)
Step 6	[CP] Wait 1s	
Step 7	[SWC] Call NvM_GetErrorStatus() for Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 8	[SWC] Fill Pim Data2 with Data_2 value	
Step 9	[SWC] Call Nvm_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_NOT_OK
Step 10		[SWC] Check that DEM Event NVM_E_WRITE_PROTECTED has been stored
Step 11	[SWC] Fill Pim Data2 with value 0xFFFF	
Step 12	[SWC] Execute NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 13	[CP] WAIT 1s	
Step 14	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 15	[SWC] Check Pim Data2 value	[SWC] value of Pim Data2 should be equal to Data_1
Step 16	[SWC] Execute NvM_SetBlockProtection() to disable write protection on Block_1	[SWC] NvM_SetBlockProtection() should return E_OK
Step 17	[SWC] Fill Pim Data1 with Data_2 value	
Step 18	[SWC]	[SWC]

	Execute NvM_WriteBlock() on Block_1	NvM_WriteBlock() should return E_OK
Step 19	[CP] WAIT 1s	
Step 20	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 21	[SWC] Fill Pim Data1 with value 0xFFFF	
Step 22	[SWC] Execute NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 23	[CP] WAIT 1s	
Step 24	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 25	[SWC] Read Pim Data1 value	[SWC] Data in Pim Data1 should be equal to Data_2
Post-conditions	None	

3.3.4 [ATS_MEM_00199] Read, write and erase DataSet block

Test Objective	Read, write and erase DataSet block		
ID	ATS_MEM_00199	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00035		
Trace to SWS Item	NVRAMManager: SWS_NvM_00006 NVRAMManager: SWS_NvM_00698 NVRAMManager: SWS_NvM_00699 NVRAMManager: SWS_NvM_00734		
Requirements / Reference to Test Environment	UC06.05.01, UC06.05.03		
Configuration Parameters	SWC uses 1 DataSet Block configured to have immediate priority (required for successful erase operation). Details are given in ATS Specification chapter 3.1.2.2.4 AT-199. For better readability, the DataSetBlockx will be replaced in this Test case by Block_x. Data_1 value =0x55AA Data_2 value =0xAA55		

Summary	The aim of this test is to verify the management of DATASET block by - Writing different data in two different data index of an NvBlocks (and switch-off/ switch-on the power) - Read data in each data index NvBlock - Erase and read data on each data index NvBlock For better readability DataIndex parameter value x of API NvM_SetDataIndex will be called DataIndex_x in this test case.	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with the initial value Data_1 value	
Step 2	[SWC] Call NvM_SetDataIndex() to set index position Block_1 to DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 3	[SWC] Call NvM_WriteBlock() for Block1	[SWC] NvM_WriteBlock() should return E_OK
Step 4	[CP] WAIT 1s	
Step 5	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 6	[SWC] Fill Pim Data1 with Data_2 value	
Step 7	[SWC] Execute NvM_SetDataIndex() to set index position of Block_1 to DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 8	[SWC] Execute NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 9	[CP] WAIT 1s	
Step 10	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 11	[LT<Power Supply>] Power off ECU	
Step 12	[CP] Wait n seconds (n depends on time needed for platform to shutdown)	
Step 13	[LT<Power Supply>] Power on ECU	
Step 14	[SWC] Write 0xFFFF value in Pim Data1	

Step 15	[SWC] Call NvM_SetDataIndex() to set index position of Block_1 to DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 16	[SWC] Execute NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 17	[CP] WAIT 1s	
Step 18	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 19	[SWC] Check Pim Data1	[SWC] Value of Pim Data1 should be equal to Data_1 value
Step 20	[SWC] Call NvM_SetDataIndex() to set index position of Block_1 to DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 21	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 22	[CP] WAIT 1s	
Step 23	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 24	[SWC] Check Pim Data1	[SWC] Data in Pim Data1 should equal to Data_2 value
Step 25	[SWC] Call NvM_EraseNvBlock() on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 26	[CP] WAIT 2s	
Step 27	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 28	[SWC] Call NvM_SetDataIndex() to set index position of Block_1 to DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 29	[SWC] Call NvM_EraseNvBlock() on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 30	[CP] WAIT 2s	
Step 31	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 32	[SWC]	[SWC]

	Call NvM_ReadBlock() for Block_1	NvM_ReadBlock should return E_OK
Step 33	[CP] WAIT 1s	
Step 34	[SWC] Call NvM_GetErrorStatus() for Block1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR
Step 35	[SWC] Call NvM_SetDataIndex() to set index position of Block_1 to DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 36	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 37	[CP] WAIT 1s	
Step 38	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR
Post-conditions	None	

3.3.5 [ATS_MEM_00202] Job notification on Native block

Test Objective	Job notification on Native block		
ID	ATS_MEM_00202	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00033		
Trace to SWS Item	NVRAMManager: SWS_NvM_00175 NVRAMManager: SWS_NvM_00176		
Requirements / Reference to Test Environment	UC06.05.11, UC06.05.12, UC06.05.13		
Configuration Parameters	SWC uses 1 Native Blocks configured to have immediate priority (required for successful erase operation). Details are given in ATS Specification chapter 3.1.2.2.5 AT-202. For readability, the NvBlockx will be replaced in this Test case by Block_x. Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the end Notification is called for block configured for		

	it, and not called when a block don't configure it. - JobEndNotification is called on each job finished with success, and JobErrorNotification is called on each job finished with error. (This is applicable for NvBlock configured with a NvMSingleCallBack) - No notification called on job finished (with or without success) for block configured with NO NvMSingleBlockCallback. Use GetErrorStatus to get the current job status (Pending, Ok, Nok...) The test should be done on each type of job (Read, Write, erase...)
Needed Adaptation to other Releases	None
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with the Data_1 value	
Step 2	[SWC] Call NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 3	[CP] WAIT 1s	
Step 4	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 5	[SWC] Fill Pim Data1 with the value 0xFFFF	
Step 6	[SWC] Execute NvM_ReadBlock() on Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 7	[CP] WAIT 1s	
Step 8	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 9	[SWC] Check Pim Data1	[SWC] Value in Pim Data1 should be equal to Data_1 value
Step 10	[SWC] Execute NvM_EraseNvBlock() on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 11	[CP] WAIT 1s	
Step 12	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId

		value 0x9 and job_result NVM_REQ_OK.
Post-conditions	None	

3.3.6 [ATS_MEM_00203] Explicit restore default values on Native block

Test Objective	Explicit restore default values on Native block		
ID	ATS_MEM_00203	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00032		
Trace to SWS Item	NVRAMManager: SWS_NvM_00171 NVRAMManager: SWS_NvM_00391 NVRAMManager: SWS_NvM_00392 NVRAMManager: SWS_NvM_00700 NVRAMManager: SWS_NvM_00456 NVRAMManager: SWS_NvM_00734		
Requirements / Reference to Test Environment	UC06.05.24		
Configuration Parameters	Need 3 Native blocks Details are given in ATS Specification chapter 3.1.2.2.6 AT-203. For more readability, NvBlockx is called Block_x in this test case. Block_1: 1- ROM Block is configured (Default values are get on the ROM address) nRomBlocks =1 Block2: 2- NvMInitBlockCallback is configured (Default values are get by a callback) nRomBlocks = 1 Block3: 3- None is configured (no callback) nRomBlocks = 0 Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the explicit restoration of default value. A write is done on a block (with value different to the configured default values), then a read check the written values. After that, a restore request is done. This is being verified by checking that the configured default values have been restored. The test must be done for different blocks where:		

	1- ROM Block is configured (Default values are get on the ROM address) 2- NvMInitBlockCallback is configured (Default values are get by a callback) 3- None is configured (No default block is loaded, block is returned invalid)	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data_1 value	
Step 2	[SWC] Call NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 3	[CP] WAIT 1s	
Step 4	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 5	[SWC] Fill Pim Data1 value with 0xFFFF	
Step 6	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 7	[CP] WAIT 1s	
Step 8	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 9	[SWC] Check Pim Data1 value	[SWC] Pim Data1 value shall be equal to Data_1 value (retrieved from NV Block)
Step 10	[SWC] Call NvM_RestoreBlockDefaults() for Block_1	[SWC] NvM_RestoreBlockDefaults() should return E_OK
Step 11	[CP] WAIT 1s	
Step 12	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 13	[SWC] Check Pim Data1 value	[SWC] Pim Data1 value shall be equal to the default value (0x1234)
Step 14	[SWC] Fill Pim Data2 with Data_1 value value	
Step 15	[SWC] Call NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_OK

Step 16	[CP] WAIT 1s	
Step 17	[SWC] Call NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 18	[SWC] Fill Pim Data2 value with 0xFFFF	
Step 19	[SWC] Call NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 20	[CP] WAIT 1s	
Step 21	[SWC] Call NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 22	[SWC] Check Pim Data2 value	[SWC] Pim Data2 value shall be equal to Data_1 value
Step 23	[SWC] Call NvM_RestoreBlockDefaults() for Block_2	[SWC] NvM_RestoreBlockDefaults() should return E_OK Check that NvMInitBlockCallback() server operation is called
Step 24	[CP] WAIT 1s	
Step 25	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 26	[SWC] Check Pim Data2 value	[SWC] Pim Data2 value should be equal to default value (0x1234)
Step 27	[SWC] Fill Pim Data3 with Data_1 value	
Step 28	[SWC] Call NvM_WriteBlock() for Block_3	[SWC] NvM_WriteBlock() should return E_OK
Step 29	[CP] WAIT 1s	
Step 30	[SWC] Call NvM_GetErrorStatus() on Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 31	[SWC] Call NvM_ReadBlock() for Block_3	[SWC] NvM_ReadBlock() should return E_OK
Step 32	[CP] WAIT 1s	
Step 33	[SWC] Call NvM_GetErrorStatus() on Block_3	[SWC] NvM_GetErrorStatus should return

		value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 34	[SWC] Check Pim Data3 value	[SWC] Pim Data3 value shall be equal to Data_1 value
Step 35	[SWC] Call NvM_RestoreBlockDefaults() for Block_3	[SWC] NvM_RestoreBlockDefaults() should return E_NOT_OK
Step 36	[CP] WAIT 1s	
Step 37	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK (the same as in step 33, since there was no other async. request accepted for this block since then)
Post-conditions	None	

3.3.7 [ATS_MEM_00254] Explicit restore default values on Redundant block

Test Objective	Explicit restore default values on Redundant block		
ID	ATS_MEM_00254	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00032		
Trace to SWS Item	NVRAMManager: SWS_NvM_00171 NVRAMManager: SWS_NvM_00391 NVRAMManager: SWS_NvM_00392 NVRAMManager: SWS_NvM_00700 NVRAMManager: SWS_NvM_00456 NVRAMManager: SWS_NvM_00734		
Requirements / Reference to Test Environment	UC06.05.24		
Configuration Parameters	Need 3 Redundant Blocks. Detailed configuration is given in ATS Specification chapter 3.1.2.2.12 AT-254. For more readability, the RedundantBlockx is called Block_x in this test case. Block_1: 1- ROM Block is configured (Default values are get on the ROM address) nRomBlocks =1 Block2: 2- NvMInitBlockCallback is configured (Default values are get by a callback) nRomBlocks = 1		

	Block3: 3- None is configured (no callback) nRomBlocks = 0 Data_1 value =0x55AA Data_2 value =0xAA55
Summary	The aim of this test is to verify the explicit restoration of default value on Redundant block. A write is done on a block (with value different to the configured default values), then a read check the written values. After that, a restore request is done. This is being verified by checking that the configured default values have been restored. The test must be done for different blocks where: 1- ROM Block is configured (Default values are get on the ROM address) 2- NvMInitBlockCallback is configured (Default values are get by a callback) 3- None is configured (No default block is loaded, block is returned invalid)
Needed Adaptation to other Releases	None
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data_1 value value	
Step 2	[SWC] Call NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 3	[CP] WAIT 1s	
Step 4	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 5	[SWC] Fill Pim Data1 value with 0xFFFF	
Step 6	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 7	[CP] WAIT 1s	
Step 8	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 9	[SWC] Check Pim Data1 value	[SWC] Pim Data1 value shall be equal to Data_1 value (retrieved from NV Block)
Step 10	[SWC]	[SWC]

	Call NvM_RestoreBlockDefaults() for Block_1	NvM_RestoreBlockDefaults() should return E_OK
Step 11	[CP] WAIT 1s	
Step 12	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 13	[SWC] Check Pim Data1 value	[SWC] Pim Data1 value shall be equal to the defaultvalue (0x1234)
Step 14	[SWC] Fill Pim Data2 with Data_1 value	
Step 15	[SWC] Call NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_OK
Step 16	[CP] WAIT 1s	
Step 17	[SWC] Call NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 18	[SWC] Fill Pim Data2 value with 0xFFFF	
Step 19	[SWC] Call NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 20	[CP] WAIT 1s	
Step 21	[SWC] Check Pim Data2 value	[SWC] Pim Data2 value shall be equal to Data_1 value
Step 22	[SWC] Call NvM_RestoreBlockDefaults() for Block_2	[SWC] NvM_RestoreBlockDefaults() should return E_OK Check that NvMInitBlockCallback() server operation is called
Step 23	[CP] WAIT 1s	
Step 24	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 25	[SWC] Check Pim Data2 value	[SWC] Pim Data2 value should be equal to default value (0x1234)
Step 26	[SWC] Fill Pim Data3 with Data_1 value	
Step 27	[SWC] Call NvM_WriteBlock() for Block_3	[SWC] NvM_WriteBlock() should return E_OK
Step 28	[CP]	

	WAIT 1s	
Step 29	[SWC] Call NvM_GetErrorStatus() on Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 30	[SWC] Fill Pim Data3 value with 0xFFFF	
Step 31	[SWC] Call NvM_ReadBlock() for Block_3	[SWC] NvM_ReadBlock() should return E_OK
Step 32	[CP] WAIT 1s	
Step 33	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 34	[SWC] Check Pim Data3 value	[SWC] Pim Data3 value should be equal to Data_1 value
Step 35	[SWC] Call NvM_RestoreBlockDefaults() for Block_3	[SWC] NvM_RestoreBlockDefaults() should return E_NOT_OK
Step 36	[CP] WAIT 1s	
Step 37	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK (the same as in step 33, since there was no other async. request accepted for this block since then)
Post-conditions	None	

3.3.8 [ATS_MEM_00255] Explicit restore default values on Dataset block

Test Objective	Explicit restore default values on Dataset block		
ID	ATS_MEM_00255	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00032		
Trace to SWS Item	NVRAMManager: SWS_NvM_00171 NVRAMManager: SWS_NvM_00391 NVRAMManager: SWS_NvM_00392		

	NVRAMManager: SWS_NvM_00700 NVRAMManager: SWS_NvM_00456 NVRAMManager: SWS_NvM_00353 NVRAMManager: SWS_NvM_00734
Requirements / Reference to Test Environment	UC06.05.24
Configuration Parameters	Need 2 DataSet blocks default values in RomBlocks = 0x1234 Detailed configuration in ATS Specification chapter 3.1.2.2.13 AT-255. For better readability, DataSetBlockx is called Blockx in this test case. Block_1 and Block_2: ROM Block is configured (Default values are copied from the ROM address) nRomBlocks =2
Summary	The aim of this test is to verify the explicit restoration of default values on Dataset blocks. A restore request is done while the dataset index point to a NV block. It is expected that this request will be rejected. A restore request is done while the dataset index point to a ROM block. It is expected that ROM data is being copied to the RAM block. For better readability *DataIndex* parameter value x of will be called DataIndex_x in this test case.
Needed Adaptation to other Releases	None
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with 0xFFFF value	
Step 2	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 3	[SWC] Call NvM_RestoreBlockDefaults() for Block_1	[SWC] NvM_RestoreBlockDefaults() should return E_NOT_OK (because DataIndex_0 points to NV block)
Step 4	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_2	[SWC] NvM_SetDataIndex() should return E_OK
Step 5	[SWC] Call NvM_RestoreBlockDefaults() for Block_1	[SWC] NvM_RestoreBlockDefaults() should return E_OK (because DataIndex_2 points to ROM block)
Step 6	[CP] WAIT 1s	
Step 7	[SWC] Call NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 8	[SWC] Check Pim Data1 value	[SWC] Pim Data1 value should be equal to

		0x1234 (default data in ROM)
Step 9	[SWC] Fill Pim Data2 with value 0xFFFF	
Step 10	[SWC] Call NvM_SetDataIndex() for Block_2 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 11	[SWC] Call NvM_RestoreBlockDefaults() for Block_2	[SWC] NvM_RestoreBlockDefaults() should return E_NOT_OK (because DataIndex_0 points to NV block)
Step 12	[SWC] Call NvM_SetDataIndex() for Block_2 and DataIndex_2	[SWC] NvM_SetDataIndex() should return E_OK
Step 13	[SWC] Call NvM_RestoreBlockDefaults() for Block_2	[SWC] NvM_RestoreBlockDefaults() should return E_OK (because DataIndex_2 points to ROM block)
Step 14	[CP] WAIT 1s	
Step 15	[SWC] Call NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 16	[SWC] Check Pim Data2 value	[SWC] Pim Data2 value should be equal to 0x1234
Post-conditions	None	

3.3.9 [ATS_MEM_00250] Write protection on Redundant Block

Test Objective	Write protection on Redundant Block		
ID	ATS_MEM_00250	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00034		
Trace to SWS Item	NVRAMManager: SWS_NvM_00325 NVRAMManager: SWS_NvM_00578 NVRAMManager: SWS_NvM_00734		
Requirements / Reference to Test Environment	UC06.05.21,		
Configuration Parameters	SWC uses 2 Redundant Blocks. Detailed configuration is described in ATS Specification at chapter 3.1.2.2.8 AT-250. For more readability, RedundantBlockx is replaced by Block_x in this test case.		

	Data_1 value =0x55AA Data_2 value =0xAA55	
Summary	The aim of this test is to verify the write block protection: • A write request is done to a write-protected block. This is expected to be rejected. • A write request is done to a WriteOnlyOnce block (that has never been written before). This is expected to be accepted and processed correctly. • A second write request is done to a WriteOnlyOnce block. This is expected to be rejected. • The write protection is deactivated for the write-protected block. • A write request is done to the formerly write-protected block. This is expected to be accepted and processed correctly.	
Needed Adaptation to other Releases	None	
Pre-conditions	The WriteOnlyOnce block has never been written before.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with the Data_1 value	
Step 2	[SWC] Execute NvM_WriteBlock() for Block1	[SWC] NvM_WriteBlock() should return E_NOT_OK (as block is write protected)
Step 3	[SWC] Check status of DEM Event NVM_E_WRITE_PROTECTED.	[SWC] Status Bit "TestFailed" of DEM Event NVM_E_WRITE_PROTECTED is set.
Step 4	[SWC] Fill Pim Data2 with the Data_1 value	
Step 5	[SWC] Call NvM WriteBlock() for Block2	[SWC] NvM_WriteBlock() should return E_OK (First write in Block should succeed)
Step 6	[CP] Wait 2s	
Step 7	[SWC] Call NvM_GetErrorStatus()	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 8	[SWC] Fill Pim Data2 with Data_2 value	
Step 9	[SWC] Call Nvm_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_NOT_OK
Step 10		[SWC] Checks that DEM Event NVM_E_WRITE_PROTECTED has been stored
Step 11	[SWC] Fill Pim Data2 with value 0xFFFF	
Step 12	[SWC] Execute NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return

		E_OK
Step 13	[CP] WAIT 1s	
Step 14	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 15	[SWC] Check Pim Data2 value	[SWC] value of Pim Data2 should be equal to Data_1
Step 16	[SWC] Execute NvM_SetBlockProtection() to disable write protection on Block1	[SWC] NvM_SetBlockProtection() should return E_OK
Step 17	[SWC] Fill Pim Data1 with Data_2 value	
Step 18	[SWC] Execute NvM_WriteBlock() on Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 19	[CP] WAIT 2s	
Step 20	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 21	[SWC] Fill Pim Data1 with value 0xFFFF	
Step 22	[SWC] Execute NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 23	[CP] WAIT 1s	
Step 24	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 25	[SWC] Check Pim Data1 value	[SWC] value of Pim Data1 should be equal to Data_2
Post-conditions	None	

3.3.10 [ATS_MEM_00251] Write protection on DataSet Block

Test Objective	Write protection on DataSet Block		
ID	ATS_MEM_00251	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement	ATR: ATR_ATR_00034		

on Acceptance Test Document		
Trace to SWS Item	NVRAMManager: SWS_NvM_00325 NVRAMManager: SWS_NvM_00578 NVRAMManager: SWS_NvM_00734	
Requirements / Reference to Test Environment	UC06.05.01,	
Configuration Parameters	SWC uses 2 Dataset Blocks Detailed configuration is available in ATS Specification chapter 3.1.2.2.9 AT-251. For more readability, WPDataSetBlockx is replaced by Block_x in this test case. Data_1 value =0x55AA Data_2 value =0xAA55	
Summary	The aim of this test is to verify the write block protection: • A write request is done to a write-protected block. This is expected to be rejected. • A write request is done to a WriteOnlyOnce block (that has never been written before). This is expected to be accepted and processed correctly. • A second write request is done to a WriteOnlyOnce block. This is expected to be rejected. • The write protection is deactivated for the write-protected block. • A write request is done to the formerly write-protected block. This is expected to be accepted and processed correctly.	
Needed Adaptation to other Releases	None	
Pre-conditions	The WriteOnlyOnce block has never been written before.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data_1 value	
Step 2	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 3	[SWC] Call NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_NOT_OK (as block is write protected)
Step 4	[SWC] Check status of DEM Event NVM_E_WRITE_PROTECTED.	[SWC] Status Bit "TestFailed" of DEM Event NVM_E_WRITE_PROTECTED is set.
Step 5	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 6	[SWC] Execute NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_NOT_OK (as block is write protected)
Step 7		[SWC]

		Check that DEM stores NVM_E_WRITE_PROTECTED default code
Step 8	[SWC] Fill Pim Data2 with the Data_1 value	
Step 9	[SWC] Call NvM_SetDataIndex() for Block_2 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 10	[SWC] Call NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_OK (First write in Block should succeed)
Step 11	[CP] WAIT 1s	
Step 12	[SWC] Execute NvM_GetErrorStatus() on Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 13	[SWC] Fill Pim Data2 with Data_2 value	
Step 14	[SWC] Call NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_NOT_OK
Step 15		[SWC] Check that DEM Event NVM_E_WRITE_PROTECTED has been stored
Step 16	[SWC] Call NvM_SetDataIndex() for Block_2 and DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 17	[SWC] Call NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_NOT_OK
Step 18		[SWC] Check that DEM has stored default code NVM_E_WRITE_PROTECTED
Step 19	[SWC] Execute NvM_SetBlockProtection() to disable write protection on Block1	[SWC] NvM_SetBlockProtection() should return E_OK
Step 20	[SWC] Fill Pim Data1 with Data_2 value	
Step 21	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 22	[SWC] Execute NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 23	[CP] WAIT 1s	
Step 24	[SWC] Execute NvM_GetErrorStatus() on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK

Step 25	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 26	[SWC] Call NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 27	[CP] WAIT 1s	
Step 28	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 29	[SWC] Fill Pim Data1 with 0xFFFF	
Step 30	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 31	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 32	[CP] WAIT 1s	
Step 33	[SWC] Check Pim Data1 value	[SWC] Pim Data1 value should be equal to Data_2
Step 34	[SWC] Fill Pim Data1 with 0xFFFF	
Step 35	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 36	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 37	[CP] WAIT 1s	
Step 38	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 39	[SWC] Check Pim Data1 value	[SWC] value of Pim Data1 should be equal to Data2
Post-conditions	None	

3.3.11 [ATS_MEM_00252] Job notification on DataSet block

Test Objective	Job notification on DataSet block		
ID	ATS_MEM_00252	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected	NvM	**State**	reviewed

Modules			
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00033		
Trace to SWS Item	NVRAMManager: SWS_NvM_00175 NVRAMManager: SWS_NvM_00176		
Requirements / Reference to Test Environment	UC06.05.01, UC06.05.02, UC06.05.03		
Configuration Parameters	SWC uses 1 DataSet Block configured to have immediate priority (required for successful erase operation). Detailed configuration is available in ATS Specification chapter 3.1.2.2.10 AT-252. For more readability, DataSetBlockx is called Block_x for this test case. Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the end Notification is called for block configured for it, and not called when a block don't configure it. - JobEndNotification is called on each job finished with success, and JobErrorNotification is called on each job finished with error. (This is applicable for NvBlock configured with a NvMSingleCallBack) The test should be done on each type of job (Read, Write, erase...) For better readability *DataIndex* parameter value x of will be called DataIndex_x in this test case.		
Needed Adaptation to other Releases	None		
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data_1 value		
Step 2	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0		[SWC] NvM_SetDataIndex() should return E_OK
Step 3	[SWC] Execute NvM_WriteBlock() for Block_1		[SWC] NvM_WriteBlock() should return E_OK
Step 4	[CP] WAIT 1s		
Step 5	[SWC] Check call of the NvMNotifyJobFinished		[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 6	[SWC] Fill Pim Data1 with Data_2 value		
Step 7	[SWC]		[SWC]

	Call NvM_SetDataIndex() for Block_1 and DataIndex_1	NvM_SetDataIndex() should return E_OK
Step 8	[SWC] Call NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 9	[CP] WAIT 1s	
Step 10	[SWC] Check call of the NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 11	[SWC] Fill Pim Data1 with value 0xFFFF	
Step 12	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 13	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 14	[CP] WAIT 1s	
Step 15	[SWC] Check call of the NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 16	[SWC] Check Pim Data1	[SWC] Value in Pim Data1 should be equal to Data_1 value
Step 17	[SWC] Fill Pim Data1 with value 0xFFFF	
Step 18	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_1	[SWC] NvM_SetDataIndex() should return E_OK
Step 19	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 20	[CP] WAIT 1s	
Step 21	[SWC] Check call of the NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 22	[SWC] Check Pim Data1	[SWC] Value in Pim Data1 should be equal to Data_2 value
Step 23	[SWC] Execute NvM_EraseNvBlock on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 24	[CP]	

	WAIT 1s	
Step 25	[SWC] Check call of the NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x9 and job_result NVM_REQ_OK.
Step 26	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_0	[SWC] NvM_SetDataIndex() should return E_OK
Step 27	[SWC] Execute NvM_EraseNvBlock on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 28	[CP] WAIT 1s	
Step 29	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x9 and job_result NVM_REQ_OK.
Post-conditions	None	

3.3.12 [ATS_MEM_00253] Job notification on Redundant block

Test Objective	Job notification on Redundant block		
ID	ATS_MEM_00253	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00033		
Trace to SWS Item	NVRAMManager: SWS_NvM_00175 NVRAMManager: SWS_NvM_00176		
Requirements / Reference to Test Environment	UC06.05.21, UC06.05.22, UC06.05.23		
Configuration Parameters	SWC uses 1 Redundant Block configured to have immediate priority (required for successful erase operation). Detailed configuration is available in ATS Specification chapter 3.1.2.2.11 AT-253. For more readability, RedundantBlockx is called Block_x in this test case. Data_1 value =0x55AA Data_2 value =0xAA55 The SWC has to provide runnables for the NvM_NotifyJobFinished callbacks and must be able to check whether one of these runnables has been invoked.		
Summary	The aim of this test is to verify the end Notification is called for block configured for		

	it, and not called when a block don't configure it. - JobEndNotification is called on each job finished with success, and JobErrorNotification is called on each job finished with error. (This is applicable for NvBlock configured with a NvMSingleCallBack) The test should be done on each type of job (Read, Write, erase...)	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with Data_1 value	
Step 2	[SWC] Call NvM Service WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 3	[CP] Wait 2s	
Step 4	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 5	[SWC] Fill Pim Data1 with value 0xFFFF	
Step 6	[SWC] Execute NvM_ReadBlock() on Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 7	[CP] Wait 1s	
Step 8	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 9	[SWC] Check Pim Data1	[SWC] Value in Pim Data1 should be equal to Data_1 value
Step 10	[SWC] Call NvM_EraseNvBlock on Block_1	[SWC] NvM_EraseNvBlock() should return E_OK
Step 11	[CP] WAIT 2s	
Step 12	[SWC] Check call of NvMNotifyJobFinished	[SWC] NvMNotifyJobFinished server operation of RUN_Ctrl has been called for Block_1 with ServiceId value 0x9 and job_result NVM_REQ_OK.
Post-conditions	None	

3.3.13 [ATS_MEM_00204] Implicit restore default values on Native Block

Test Objective	Implicit restore default values on Native Block		
ID	ATS_MEM_00204	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00032		
Trace to SWS Item	NVRAMManager: SWS_NvM_00171 NVRAMManager: SWS_NvM_00202 NVRAMManager: SWS_NvM_00657		
Requirements / Reference to Test Environment	UC06.05.24		
Configuration Parameters	SWC uses 3 Native Blocks configured to have immediate priority (required for successful erase operation. Detailed configuration is available in ATS Specification chapter 3.1.2.2.7 AT-204. For more readability, NvBlockx is called Block_x in this test case. Block_1: 1- ROM Block is configured (default value 0x1234 is copied from the ROM address) nRomBlocks =1 Block2: 2- NvMInitBlockCallback is configured (default value 0x5678 is copied by a callback) nRomBlocks = 1 Block3: 3- None is configured (no callback) nRomBlocks = 0 3 Per Instance Memories defined: Block1_Data, Block2_Data, Block3_Data Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the implicit restoration of default value. Default values are loaded after a bad read The test must be done for different blocks where: 1- ROM Block is configured (Default values are get on the ROM address) 2- NvMInitBlockCallback is configured (Default values are get by a callback) 3- None is configured (No default block is loaded, block is returned invalid)		
Needed Adaptation to other Releases	None		
Pre-conditions	None		
Main Test Execution			
Test Steps		**Pass Criteria**	

Step 1	[SWC] Fill Pim Data1 with the Data_1 value	
Step 2	[SWC] Execute NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 3	[CP] WAIT 1s	
Step 4	[SWC] Execute NvM_GetErrorStatus on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 5	[SWC] Call NvM_EraseNvBlock() for Block_1	[SWC] NvM_EraseBlock() shall return E_OK
Step 6	[CP] WAIT 1s	
Step 7	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 8	[SWC] Fill Pim Data1 with the value 0xFFFF	
Step 9	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 10	[CP] WAIT 1s	
Step 11	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_RESTORED_FROM_ROM
Step 12	[SWC] Check Pim Data1	[SWC] Pim Data1 should be equal to default value (0x1234)
Step 13	[SWC] Fill Pim Data2 with the Data_2 value	
Step 14	[SWC] Execute NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_OK
Step 15	[CP] WAIT 1s	
Step 16	[SWC] Call NvM_GetErrorStatus() for Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 17	[SWC] Call NvM_EraseNvBlock() for Block_2	[SWC] Check that NvM_EraseBlock() returns E_OK
Step 18	[CP] Wait 1s	
Step 19	[SWC] Call NvM_GetErrorStatus() for Block_2	[SWC] NvM_GetErrorStatus should return

		value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 20	[SWC] Fill Pim Data2 with the value 0xFFFF	
Step 21	[SWC] Call NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 22	[CP] WAIT 1s	
Step 23	[SWC] Check call of NvM_NotifyInitBlock() server operation for Block_2	[SWC] NvM_NotifyInitBlock() server operation for Block_2 is called for restoring the ROM DEFAULT value
Step 24	[SWC] Call NvM_GetErrorStatus() for Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_RESTORED_FROM_ROM.
Step 25	[SWC] Check Pim Data2	[SWC] Pim Data2 should be equal to default value (0x5678)
Step 26	[SWC] Fill Pim Data3 with the Data_1 value	
Step 27	[SWC] Execute NvM_WriteBlock() for Block_3	[SWC] NvM_WriteBlock() shall return E_OK
Step 28	[CP] WAIT 1s	
Step 29	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 30	[SWC] Call NvM_EraseNvBlock() for Block_3	[SWC] NvM_EraseNvBlock() should return E_OK
Step 31	[CP] WAIT 1s	
Step 32	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 33	[SWC] Fill Pim Data3 with the value 0XFFFF	
Step 34	[SWC] Call NvM_ReadBlock() for Block_3	[SWC] NvM_ReadBlock() should return E_OK
Step 35	[CP] WAIT 1s	

Step 36	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR
Step 37	[SWC] Check Pim Data3	[SWC] Pim Data3 should be equal to value 0xFFFF
Post-conditions	None	

3.3.14 [ATS_MEM_00256] Implicit restore default values on Redundant Block

Test Objective	Implicit restore default values on Redundant Block		
ID	ATS_MEM_00256	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00032		
Trace to SWS Item	NVRAMManager: SWS_NvM_00171 NVRAMManager: SWS_NvM_00202 NVRAMManager: SWS_NvM_00657		
Requirements / Reference to Test Environment	UC06.05.24		
Configuration Parameters	Need 3 Redundant Blocks configured to have immediate priority (required for successful erase operation). Detailed configuration is available in ATS Specification 3.1.2.2.14 AT-256. For more readability RedundantBlockx is called Blockx in this test case. Block_1: 1- ROM Block is configured (Default value 0x1234 is copied from the ROM address) nRomBlocks =1 Block2: 2- NvMInitBlockCallback is configured (Default value 0x5678 is copied by a callback) nRomBlocks = 1 Block3: 3- None is configured (no callback) nRomBlocks = 0 3 Per Instance Memories defined: Block1_Data, Block2_Data, Block3_Data Data_1 value =0x55AA Data_2 value =0xAA55		
Summary	The aim of this test is to verify the implicit restoration of default value on a Redundant block.		

	Default values are loaded after a bad read. The test must be done for different blocks where: 1- ROM Block is configured (Default values are get on the ROM address) 2- NvMInitBlockCallback is configured (Default values are get by a callback) 3- None is configured (No default block is loaded, block is returned invalid)	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with the Data_1 value	
Step 2	[SWC] Execute NvM_WriteBlock() for Block_1	[SWC] NvM_WriteBlock() should return E_OK
Step 3	[CP] WAIT 2s	
Step 4	[SWC] Execute NvM_GetErrorStatus on Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 5	[SWC] Call NvM_EraseNvBlock() for Block_1	[SWC] NvM_EraseBlock() shall return E_OK
Step 6	[CP] WAIT 2s	
Step 7	[SWC] Call NvM_GetErrorStatus()for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 8	[SWC] Fill Pim Data1 with the value 0xFFFF	
Step 9	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 10	[CP] WAIT 1s	
Step 11	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_RESTORED_FROM_ROM
Step 12	[SWC] Check Pim Data1	[SWC] Pim Data1 should be equal to default value (0x1234)
Step 13	[SWC] Fill Pim Data2 with the Data_2 value	
Step 14	[SWC] Execute NvM_WriteBlock() for Block_2	[SWC] NvM_WriteBlock() should return E_OK
Step 15	[CP]	

	WAIT 2s	
Step 16	[SWC] Call NvM_GetErrorStatus() for Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 17	[SWC] Call NvM_EraseNvBlock() for Block_2	[SWC] Check that NvM_EraseBlock() returns E_OK
Step 18	[CP] Wait 2s	
Step 19	[SWC] Call NvM_GetErrorStatus()	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 20	[SWC] Fill Pim Data2 with the value 0xFFFF	
Step 21	[SWC] Call NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 22	[CP] WAIT 1s	
Step 23	[SWC] Check call of NvM_NotifyInitBlock() server operation for Block_2	[SWC] NvM_NotifyInitBlock() server operation for Block_2 is called for restoring the ROM DEFAULT values
Step 24	[SWC] Call NvM_GetErrorStatus() for Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_RESTORED_FROM_ROM.
Step 25	[SWC] Check Pim Data2	[SWC] Pim Data2 should be equal to default value (0x5678)
Step 26	[SWC] Fill Pim Data3 with the Data_1 value	
Step 27	[SWC] Execute NvM_WriteBlock() for Block_3	[SWC] NvM_WriteBlock() shall return E_OK
Step 28	[CP] WAIT 2s	
Step 29	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 30	[SWC] Call NvM_EraseNvBlock() for Block_3	[SWC] NvM_EraseNvBlock() should return E_OK
Step 31	[CP] WAIT 2s	
Step 32	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should

		be equal to NVM_REQ_OK
Step 33	[SWC] Fill Pim Data3 with the value 0XFFFF	
Step 34	[SWC] Call NvM_ReadBlock() for Block_3	[SWC] NvM_ReadBlock() should return E_OK
Step 35	[CP] WAIT 1s	
Step 36	[SWC] Call NvM_GetErrorStatus() for Block_3	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_INTEGRITY_FAILED, but is not strictly mandated by AUTOSAR
Step 37	[SWC] Check Pim Data3	[SWC] Pim Data3 should be equal to value 0xFFFF
Post-conditions	None	

3.3.15 [ATS_MEM_00257] Implicit restore default values on Dataset Block

Test Objective	Implicit restore default values on Dataset Block		
ID	ATS_MEM_00257	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00032		
Trace to SWS Item	NVRAMManager: SWS_NvM_00171 NVRAMManager: SWS_NvM_00354		
Requirements / Reference to Test Environment	UC06.05.24		
Configuration Parameters	Need 2 DataSet blocks default values in RomBlocks = 0x1234 Detailed configuration is available in ATS Specification chapter 3.1.2.2.15 AT-257. For more readability, DataSetBlockx is called Block_x in this test case. Block_1: 1- ROM Block is configured (Default values are get on the ROM address) nRomBlocks =2 Block2: 2- NvMInitBlockCallback is configured (Default values are get by a callback) nRomBlocks = 2		
Summary	The aim of this test is to verify the implicit restoration of default value on a Dataset block. For Dataset blocks this can be done via a read request while the dataset index point. to a ROM block. It is expected that ROM data is being copied to the RAM		

	block. For better readability *DataIndex* parameter value x of will be called DataIndex_x in this test case.	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Fill Pim Data1 with the value 0xFFFF	
Step 2	[SWC] Call NvM_SetDataIndex() for Block_1 and DataIndex_2	[SWC] NvM_SetDataIndex() should return E_OK
Step 3	[SWC] Call NvM_ReadBlock() for Block_1	[SWC] NvM_ReadBlock() should return E_OK
Step 4	[CP] WAIT 1s	
Step 5	[SWC] Call NvM_GetErrorStatus() for Block_1	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 6	[SWC] Check Pim Data1	[SWC] Pim Data1 should be equal to default value (0x1234)
Step 7	[SWC] Fill Pim Data2 with the value 0xFFFF	
Step 8	[SWC] Call NvM_SetDataIndex() for Block_2 and DataIndex_3	[SWC] NvM_SetDataIndex() should return E_OK
Step 9	[SWC] Call NvM_ReadBlock() for Block_2	[SWC] NvM_ReadBlock() should return E_OK
Step 10	[CP] WAIT 1s	
Step 11	[SWC] Call NvM_GetErrorStatus()for Block_2	[SWC] NvM_GetErrorStatus should return value E_OK and the RequestResult parameter value should be equal to NVM_REQ_OK
Step 12	[SWC] Check Pim Data2	[SWC] Pim Data2 should be equal to default value (0x1234)
Post-conditions	None	