BMW GROUP

Rolls-Royce Motor Cars Limited

Dlog Classic Integration Manual

Project	BMW AUTOSAR 4 Core Rel. 3
Author	BMW AG
Release Date	2017-12-14
Version	5.3.1
Status	Release
Hotline	+49 89 382 - 32233
Contact	bac@bmw.de https://asc.bmw.com/jira/browse/BSUP (extern) https://asc.bmwgroup.net/jira/browse/BSUP (intern)

Revision History

Version	Date	Description
5.3.1	2017-12-14	Update NvM configuration
5.3.0	2017-11-09	Version Update
5.2.1	2017-10-12	Fix RDBI commands
5.2.0	2017-09-14	Fix RDBI commands
5.1.0	2017-08-10	Remove deprecated stuff, change FirstStartFlag to ProgId
5.0.0	2017-06-29	Initial version for SP2021

Company
Bayerische Motoren Werke Aktiengesellschaft
Postal address
BMW AG
80788 München
Office address
Forschungs- und Innovationszentrum (FIZ)
Hufelandstr. 1
80937 München
Telephone
Switchboard
+49 89 382-0
Internet
www.bmwgroup.com

Table of Contents

1 Introduction

General

For a general introduction to the BAC4/aBAC Modules please refer to [1].

This document only describes topics related to the Dlog BAC4/aBAC Module.

This Integration Manual describes the basis functionality, API and the configuration of the BMW system function Dlog.

Functional overview

The Data Logistic is part of the BootManager, Bootloader and Application. It is needed for every ECU to

- get production information
- identify software and hardware
- check compatibility of hardware and software
- check compatibility of all software units (SWEs)
- check at startup if the software is valid and may be started.

2 Acronyms and Abbreviations

API	Application Programming Interface
Application	Application stands for the high-level part of software that uses the APIs provided by the modules. It can also mean the driving application that does not belong to the Bootloader.
AUTOSAR	Automotive Open System Architecture
CCC	Car Communication Computer
Central Pia Master	Central instance controlling and managing Pia functionality (PIA-Zentralinstanz).
Coding	Coding Client
DTC	Diagnostic Trouble Code -> Fehlercode des Fehlerspeichereintrages
ECU	Electronic Control Unit
FAT	Flash-Absicherungs-Tool
FZG	Fahrzeug
HO	Handelsorganisation (BMW)
HW	Hardware
IDRL	Individual Data Recovery - light
OS	Operating System
Pia	Personalisierung, Individualisierung, Adaption (Personalization, Individualization, Adaptation)
Pia value	Scalar setting used by a Pia function.
PiaClient	Basic software module providing services for Pia functions.
Profile	Union of all personal Pia values.
RAM Block	The part of an NVRAM Block that resides in the RAM. A RAM block is used as shared memory interface between the NVRAM manager and the PiaClient.
RTE	Runtime Environment
SG	Steuergerät
SGID	Steuergeräte-ID, Diagnoseadresse, Steuergeräte-Adresse
SID	Service Identifier
SW	Software
SW-C	Software Component
UDS	Universal Diagnostic Services
VIN	Vehicle Identification Number
VIN7	The last 7 digits of the 17-digit VIN

All abbreviations used throughout this document -- except the ones listed here -- can be found in the official AUTOSAR glossary [2].

3 Related documentation

References

[1] BAC4 General Concept for the Module Integration
BAC4_General_Concepts_for_the_Module_Integration.pdf

[2] Glossary
AUTOSAR_TR_Glossary

[3] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary

[4] Specification of Diagnostic Communication Manager
AUTOSAR_SWS_DiagnosticCommunicationManager

[5] Specification of Memory Abstraction Interface
AUTOSAR_SWS_MemoryAbstractionInterface

[6] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager

[7] Specification of RTE Software
AUTOSAR_SWS_RTE

4 Limitations

Dlog has been validated on HWs where erased flash cells can be read without triggering a ECC exception. Nevertheless it should also run on other types of HWs, see section 6.3, ‘‘ECC error handling’’ for details.

5 Software Architecture

Dependencies on AUTOSAR modules

CRC Library

In the Bootmanager, the CRC Library [3] is used for checking the SWEs against data corruption.

DCM

The module Dcm [4] will call functionality of the module DataLogistic when a ReadDataByIdentifier, WriteDataByIdentifier or RoutineControl has been received for an logistic operation. In Application the corresponding R-ports of the Dcm for these operations shall be connected with the corresponding P-ports of the module DataLogistic. In Bootloader the Dcm calls the C-Api of DataLogistic.

MemIf

In Bootloader, the NV-RAM blocks are initialized via MemIf [5].

NvM

In Application, the NV-RAM blocks are initialized through the NvM [6].

RTE

Only in Application the module DataLogistic is realized as a software component and is using RTE services [7] for client/server as well as sender/receiver communication to communicate with other SWCs.

Dependencies on BMW modules

Bootmanager

In Bootmanager, the file `Bmhw_Platform_Cfg.h` is #included. It must define the function `BM_CLEAR_HARDWARE_ECC_ERROR_FLAG()`.

BUtil

The BUtil library provides utility functions used by Dlog.

Coding

The connection to the module Coding is used in Application. There are two connections between Coding and DataLogistic. The module Coding provides functions to get the CAF-IDs. The module DataLogistic provides the current programming ID for the Coding and other concerned modules.

The corresponding P-/R-ports of the module DataLogistic shall be connected with the corresponding R-/P-ports of the module Coding.

DlogUser

If the configuration parameter `DlogShared/DlogSharedPlatform/MultiCpuEnable` is set to true, the multi cpu features of the Dlog module are enabled. In this case, the user/integrator needs to implement the functions declared in `DlogUser.h`. In the application, where the RTE is enabled, the C/S interfaces `DlogUser_Svk`, `DlogUser_DevelopmentInfo` and `DlogUser_SweProgrammingStatus` need to be provided.

If the configuration parameters `Dlog/Dlog_Features/ExceptionHandlerHandleSWEErrors` and `DlogShared/DlogSharedPlatform/HandleEccRom` are set to TRUE, the user/integrator needs to implement the function `Dlog_UserTriggerEccCheck()` (see section 6.3, "ECC error handling" for details).

The API documentation can be found in the `Dlog_User.h` header file in doxygen format.

ProgId

In the Application, there is a special P-Port `ProgId` (see section 6.4, "Configuration of the RTE"). It can be used by the user/integrator to retrieve the Programming Id, which changes after each reprogramming. So it can be used to determine whether the ECU has been reprogrammed since the last check of this Id. The Id can then be saved to NvM to check for the next reprogramming.

6 Integration

Configuration of other Modules

The following modules shall be configured, before the module Dlog is compiled and linked.

Dcm

Read Data By Identifier

The following RDBI commands shall be configured within Dcm.

Id	Length (bytes)	Connect to (RTE interface)
0x2504	12	TimingParameters
0x8000	16	EcuUid
0xf101	Variable, max=RteSvkArrayMaxSize[1]	SvkCurrent
0xf102	$4 + f + 8 * (ns + nh)^2$	SvkSystemSupplier
0xf103	$4 + f + 8 * (ns + nh)^2$	SvkPlant
0xf10[i+4]	$4 + f + 8 * (ns + nh)^2$	SvkBackup[i]
0xf152	2	HWModificationIndex
0xf18b	3	EcuManufacturingDate
0xf18c	EcuSerialNumberLength[3]	EcuSerialNumber
0xf190	17	Vin

Routine Control

The following RC command shall be configured within Dcm.

ReadDevelopmentInfoField			
Id	0x205		
Fixed Length	False		
Start Routine	Dlog_RoutineControlStartReadDevelopmentInfoField		
Input Signals	**Id**	**Pos**	**Length (bytes)**
	0	0	8 (variable)
Output Signals	**Id**	**Pos**	**Length (bytes)**
	0	0	DevelopmentInfoLength (variable)
Stop Routine	<None>		
Info Routine	<None>		

[1] Configuration parameter, see DlogClassic_paramef.arxml.
[2] Where ns the number of SWEs, nh the number of HWEs and f the length of the fingerprint.
[3] Configuration parameter, see DlogShared_paramef.arxml.

Nvm

The Dlog needs the following Nvm blocks to store the logistic data.

NVM_BLOCK_BOOT_PROG_DATA	
NvMBlockHeaderInclude	Dlog_Nvm.h
NvMBlockJobPriority	0
NvMBlockManagementType	NVM_BLOCK_NATIVE
NvMBlockWriteProt	True
NvMInitBlockCallback	-
NvMNvBlockLength	48[4]
NvMNvBlockNum	1
NvMRamBlockDataAddress	Dlog_ProgData
NvMResistantToChangedSw	true, if NvMDynamicConfiguration = true
NvMRomBlockDataAddress	-
NvMRomBlockNum	0
NvMSelectBlockForReadall	true
NvMSelectBlockForWriteAll	false
NvMSingleBlockCallback	-
NvMWriteBlockOnce	False

Write Frequency: This block will be written on

- authentication
- Write Data By Identifier VIN
- programming the ECU
- first start up after programming the ECU.

NVM_BLOCK_BOOT_SVK_HISTORY	
NvMBlockHeaderInclude	Dlog_Nvm.h
NvMBlockJobPriority	0
NvMBlockManagementType	NVM_BLOCK_NATIVE
NvMBlockWriteProt	False
NvMInitBlockCallback	-
NvMNvBlockLength	10[4]
NvMNvBlockNum	1
NvMRamBlockDataAddress	Dlog_SvkHistory
NvMResistantToChangedSw	true, if NvMDynamicConfiguration = true
NvMRomBlockDataAddress	-
NvMRomBlockNum	0
NvMSelectBlockForReadall	true
NvMSelectBlockForWriteAll	false
NvMSingleBlockCallback	-
NvMWriteBlockOnce	False

[4]block length might differ depending on the used compiler and compiler settings

Write Frequency: This block will be written after programming the ECU with a valid application.

NVM_BLOCK_DLOG_SVK_ENTRY	
NvMBlockHeaderInclude	Dlog_Nvm.h
NvMBlockJobPriority	0
NvMBlockManagementType	NVM_BLOCK_DATASET
NvMBlockWriteProt	False
NvMInitBlockCallback	-
NvMNvBlockLength	$8 + 8 *$ MaxNumberOfHistorySgbmIds
NvMNvBlockNum	$2 +$ DlogNumberOfSvkBackups
NvMRamBlockDataAddress	-
NvMResistantToChangedSw	true, if NvMDynamicConfiguration = true
NvMRomBlockDataAddress	-
NvMRomBlockNum	0
NvMSelectBlockForReadall	false
NvMSelectBlockForWriteAll	false
NvMSingleBlockCallback	-
NvMWriteBlockOnce	False

Write Frequency: This block will be written after programming the ECU with a valid application.

MemIf

When not using the Nvm, `Dlog_InitNvm()` shall be called at startup in order to read the NV blocks via MemIf. All blocks described in 0 must be configured to be used via the MemIf module.

Configuration

For details about the configuration parameters of the module Dlog please refer to the description in the Dlog*_paramdef.arxml files.

The Dlog configuration contains the following containers:

- DlogGeneral
- DlogFeatures
- DlogUser
- DlogSharedGeneral
- DlogSharedInterface
- DlogSharedPlatform

DlogGeneral

This container contains the general configuration (parameters) of the Dlog module.

DlogFeatures

This container contains the switches for enabling and disabling certain features of the Dlog module. Depending on whether the Dlog module is used in Application, Bootloader or Bootmanager, different features need to be enabled or disabled.

DlogUser

This Container contains user defined callbacks.

DlogSharedGeneral

This container can be found in the DlogShared module, which contains the configuration parameters that are shared between Application, Bootloader and Bootmanager. Therefore, there shall be only one parameter configuration file, which is then used during the generation of the Dlog module in Application, Bootloader and Bootmanager. For details on the <Module>Shared concept, please refer to [1].

For this container, the make verify rule not only verifies the configuration but also outputs information about the SWE configuration in a human readable format.

Logistic

This container contains the general logistic parameters. Special attention should be given to the choice of DlogHweTableLocation. The default is `DLOG_HWE_LOC_ROM` which puts the HWE table into a read-only memory section and should be ok in most cases. This memory section shall be placed somewhere, where it cannot be overwritten by an application or bootloader update. Note that the HWE table is read during normal operation in bootloader and application. Depending on your hardware this may interfere with the garbage collection of the Flash EEPROM Emulation for example. In such a case, the HWE table should be placed in RAM (`DLOG_HWE_LOC_RAM`). Then it is the responsibility of the integrator initialize this RAM location with the corresponding values in ROM during ECU startup.

DlogSharedInterface

This container contains options for interfaces to other modules.

DlogSharedPlatform

This container contains platform dependent functions.

ECC error handling

If the hardware supports ECC error detection, `DlogShared/DlogSharedPlatform/HandleEccRom` should be set to TRUE.

Whenever ROM (Flash) ECC errors occur, it is expected, that either

- `Dlog_RomAccessExceptionHandler()` is called directly in the context of the CPU exception, or
- the ECC error is registered and `Dlog_RomAccessExceptionHandler()` is called later but must return before the function `Dlog_UserTriggerEccCheck()` returns.

If the ECC error is not handled by `Dlog_RomAccessExceptionHandler()`, i.e. if `ExceptionHandlerHandleSWEErrors` is set to `FALSE` or the exception did not occur on a flash segment containing the valid flags of an SWE, `Dlog_UserRomAccessExceptionHandler()` is called.

Configuration of the RTE

After performing the steps indicated in section 6.1, "Configuration of other Modules" and section 6.2, "Configuration", the RTE configuration can be started. Except for connecting the ports according to the table below, no special Rte configuration is needed for Dlog.

Port[5]	**SWC / BSW**
Coding_Svk	Coding
DETService	DET
DevelopmentInfoField	DCM
DlogUser_DevelopmentInfo	User defined SWC
DlogUser_SvkCurrent	User defined SWC
DlogUser_SvkHistory	User defined SWC
DlogUser_SweProgrammingStatus	User defined SWC
EcuInfo	Any SWC who needs it
EcuSerialNumber	DCM
FlashTimingParameter	DCM
HWModificationIndex	DCM
InitBlockProgData	NvM
InitBlockSvkBackup<x>	NvM
InitBlockSvkHistory	NvM
InitBlockSvkPlant	NvM
InitBlockSvkSysSupp	NvM
LifeCycle	BswM
LifeCycleRequest	BswM
ManufacturingDate	DCM
NvMAdmin_DlogProgData	NvM
NvMNotifyJobFinished_DlogProgData	NvM
NvMService_DlogProgData	NvM
NvMService_DlogSvkEntry	NvM
NvMService_DlogSvkHistory	NvM

ProgId	Any SWC who needs it (e.g. Coding)
SgbdIdx	DCM
SvkBackup<x>	DCM
SvkCurrent	DCM
SvkPlant	DCM
SvkSysSupplier	DCM
SweInfo	Any SWC who needs it
SweSignatureAccess	Any SWC who needs it
SwfkDeleteSupported	DCM
Vin	DCM

Software Integration

Startup/Initialisation

Preconditions

Before the INIT-Mode is requested, the following preconditions shall be satisfied:

- `NvM_ReadAll` must be finished

Postconditions

- The Dlog module switches to INITIALIZED-Mode and then to RUNNING-Mode.
- The FirstStartMode switches according to the FristStartFlag

Normal Operation

Nothing to be done here.

Shutdown/Deactivation

Preconditions

Before the STOPPED-Mode is requested, the Dlog module must be in RUNNING-Mode.

Postconditions

The Dlog module switches to STOPPED-Mode.

[5]Some of these ports might be not available depending on the specific configuration of the Dlog module

Memory Mapping

The following memory sections shall be defined in their corresponding application SWEs and marked as overlay in all other SWEs:

APPL_<START|STOP>_SEC_CONST_SWE<x>_DESCRIPTION_TABLE
The description table, i.e. SGBM-ID and optional Development Info Field of SWE <x>.

APPL_<START|STOP>_SEC_CONST_SWE<x>_FLASH_STATUS
The flash status, i.e. CRC, Valid Flags, and Programming Dependency Flags of SWE <x>.

APPL_<START|STOP>_SEC_CONST_SWE<x>_SIGNATURE
The signature of SWE <x>.

The following memory sections shall be defined in the bootloader SWEs and marked as overlay in all other SWEs:

BL_<START|STOP>_SEC_CONST_BOOTSWE_DESCRIPTION_TABLE
The description table, i.e. SGBM-ID and optional Development Info Field of the boot SWE.

BL_<START|STOP>_SEC_CONST_BOOTSWE_FLASH_STATUS
The flash status, i.e. CRC, Valid Flags, and Programming Dependency Flags of the boot SWE.

BL_<START|STOP>_SEC_CONST_BOOTSWE_SIGNATURE
The signature of the boot SWE.

Dlog_<START|STOP>_SEC_CONST_SHARED_SWE_DATA
Shared Data, that is located in the Bootloader and also accessd from Bootmanager and Application.

Dlog_<START|STOP>_SEC_CONST_SHARED_MEMSEGTBL_DATA
The memory segmentation table.

The following memory sections shall be defined in the bootmanager and marked as overlay in all SWEs:

Dlog_<START|STOP>_SEC_CONST_BM_HW_DESCRIPTION_TABLE
The hardware description table, i.e. the SGBM-IDs of the HWEs, the ECU serial number and the manufacturing data.

Dlog_<START|STOP>_SEC_VAR_SHARED_SWESTATUS
The SWE status, i.e. the error flags for the SWEs

7 Post Integration

SWE Generator Config File

After the Dlog/DlogShared module is fully configured, it is possible to generate a template configuration file for the SWE Generator. This is done by

```
make generate_Dlog_sweconfig [Dlog_SWE_CONFIG_FILE=<SweConfigFile>]
    [Dlog_SWE_CFG_OUTPUT_FILE=<SweCfgOutputFile>]
```

The optional parameter `Dlog_SWE_CONFIG_FILE` defines the SWE configuration file. By default, `sample/swecfg/DlogSweCfg_1.arxml` is chosen, which is a sample configuration for the SWE number 1.

The optional parameter `Dlog_SWE_CFG_OUTPUT_FILE` defines the output file name. By default, this is `swegenerator_1.cfg`. Note, that this is only the file name, not the full path. The output path will always be `$(Dlog_OUTPUT_PATH)`.

The SWE configuration file (don't confuse it with the SWE Generator configuration file) is a simple ARXML parameter configuration file containing the following 3 parameters:

Parameter	Description
SweNr	Integer containing the SWE number for which the SWE Generator configuration file shall be written.
SgbmId	String containing the SgbmId of the SWE.
SweStatusAtEnd	Boolean defining whether the SWE Flash Status shall be placed at the end (TRUE) or at the beginning (FALSE) of the SWE.