VECTOR

Flash Bootloader

Technical Reference

Communication Wrapper PDU Router

Version 3.02.00

Authors	Jörn Herwig, Ralf Hägenläuer, Andreas Wenckebach, Achim Strobelt, Gunnar Meiss
Status	Released

Document Information

History

Author	Date	Version	Remarks
Achim Strobelt	2017-12-20	3.00.00	Cfg5 configuration module
Achim Strobelt	2018-03-12	3.01.00	Decline requests, ComM handling
Achim Strobelt	2018-05-05	3.02.00	Remove usage of ASR ComM module for CAN

Reference Documents

No.	Source	Title	Version
[1]	Vector	User Manual Flash Bootloader	2.7
[2]	Vector	Technical Reference MICROSAR PDU Router – DaVinci Configurator	3.00.00

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Introduction

This document covers the Communication Wrapper PDU Router component of the Flash Bootloader (FBL).

The component is used in Bootloader environments where a sub-set of the MICROSAR communication stack, up to the PDU router is used. Above this an additional communication wrapper layer adapts the PduR interface to the respective FBL interfaces. From the ASR point of view this wrapper (FblWrapperCom / FblCw) is implemented as a CDD with communication stack contribution.

1.1 Architecture Overview

The following figure shows the location of the Communication Wrapper in the FBL architecture.

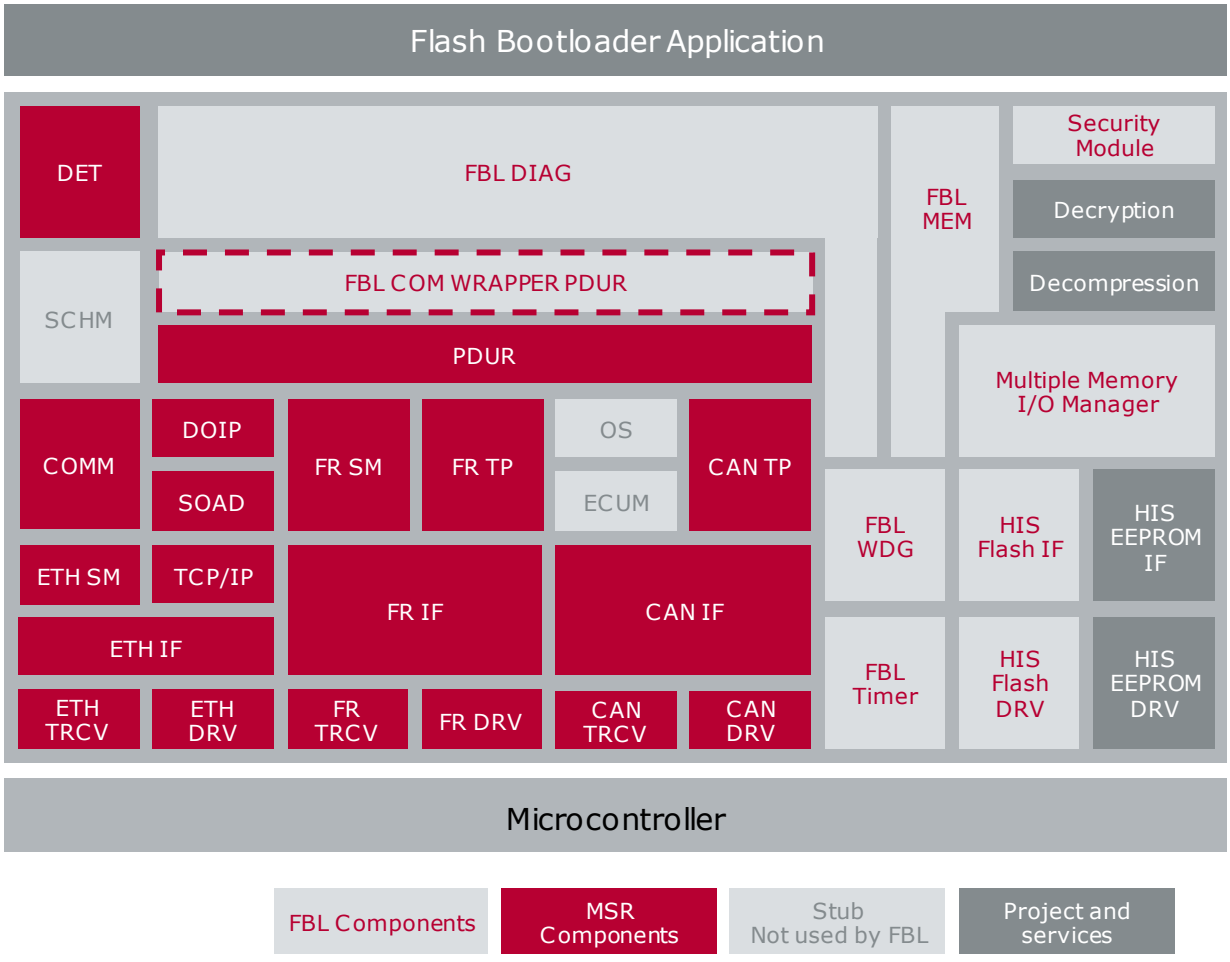

Figure 1-1 FBL Architecture Overview

2 Functional Description

The main purpose of the Communication Wrapper component is the abstraction of the particularities of a specific communication stack. It offers a specific API to the FBL diagnostics, allowing the reception and transmission of diagnostic requests and responses.

In this context, the concept of so-called connections is used. A connection represents an individual diagnostic communication entity, consisting of the following elements:

- Physical request PDUs (at least one)
- Functional request PDUs (optional, at least one if used)
- Physical response PDU

The PDUs are used to interface with a PDU router lower layer module, implementing the transport protocol API.

Requests received on the physical or functional PDU of a specific connection are passed to the Bootloader diagnostic module. The respective response (if any) is transmitted on the matching physical response PDU.

2.1 Features

The Communication Wrapper PDU Router interfaces with the MICROSAR PDU router and other relevant components of the communication stack.

Besides the basic requirements fulfilled by every Communication Wrapper variant, the Communication Wrapper PDU Router offers some additional features.

2.1.1 Multiple Bus Types

The communication isn't limited to a single bus type. Instead, any bus type available for the MICROSAR communication stack can be added on demand.

Currently, the following bus types are supported:

- CAN / ISO-TP (ISO 15765)
- Ethernet / DoIP
- Ethernet / SoAD
- FlexRay / ISO-TP (ISO-10681)

2.1.2 Multiple Connections

Multiple connections can be configured. This multiplicity isn't limited to a single channel or even bus type. Instead any valid combination of the following is possible:

- Multiple connections per channel and bus type, e.g. multiple tester instances with individual addresses
- Multiple bus types, e.g. CAN and Ethernet

Once a valid connection is established, requests from other connections are refused.

2.1.3 Baud Rate Switch

The baud rate of the channel associated with the active connection can be switched at run-time, e.g. when requested through a LinkControl service request.

The available baud rates and specific hardware parameters can be configured in DaVinci Configurator.

Note
Currently switching of the baud rates is only supported for CAN.

3 Integration

3.1 Scope of Delivery

The delivery of the Communication Wrapper PDU Router contains the files which are described in the following chapters.

3.1.1 Static Files

The first group contains components found in the BSW\FblCw folder of your delivery. Do not modify the contents of the files in this group without prior written permission from Vector (modification of these files will void your warranty).

File	Description
Fbl_cw.c/h	Communication wrapper layer for ASR4 PDU router (PduR)

Table 3-1 Core Files

3.1.2 Generated Files

The second set of files is generated by the configuration tool, DaVinci Configurator. You should not modify these files manually.

File	Description
PduR_Dcm.h	PDU router function API called by Communication Wrapper.
FblCw_Cfg.h	This file contains general configuration precompile switches of CONFIG-CLASS PRE_COMPILE data.
Dcm_Cbk.h	This is the generated header file of FblCw containing prototypes and symbolic PDU handles for lower layers, e.g. the PduR.
FblCw_Lcfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS PRE-COMPILE and CONFIG-CLASS LINK data.
FblCw_Lcfg.c	This file contains: > local constant macros > local function macros > local data types and structures > local data prototypes > local data > global data > global functions of CONFIG-CLASS PRE-COMPILE and CONFIG-CLASS LINK data.

File	Description
FblCw_PBcfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS POST-BUILD data.
FblCw_PBcfg.c	This file contains: > local constant macros > local function macros > local data types and structures > local data prototypes > local data > global data of CONFIG-CLASS POST-BUILD data.

Table 3-2 Generated Files

4 Configuration

4.1 Preparation

The additional tools “DaVinci External Components” are needed to execute the VASE script mentioned below. Please download the tools from the Vector homepage and install the component “Vector AUTOSAR Scripting Engine” and all components needed to run the scripting engine.

Reference
Version 2.7.1 can be downloaded from https://vector.com/vi_news_detail_de,,2816,1713167.html.
Newer versions can be found as service pack of DaVinci Configurator.

4.2 DaVinci Configurator

The following guide shows how to configure a bootloader project in DaVinci Configurator. The focus is put on the OEM independent characteristics.

Please see the OEM specific Technical Reference document for OEM specific settings.

Note
Please note that this chapter documents bootloader specific settings only. For other settings, please refer the MICROSAR Technical Reference documents provided with your project.

Caution
Vector Bootloaders normally run in polling mode. Polling mode must be activated in all bus components.

4.2.1 Project Creation

The creation of a DaVinci developer project must be done in several steps:

4.2.1.1 Load Network Description File

After a DaVinci Configurator project is created, the network setup must be configured with an **Input File**. These files could be Autosar ECU extract files or legacy network definitions like DBC or Fibex files.

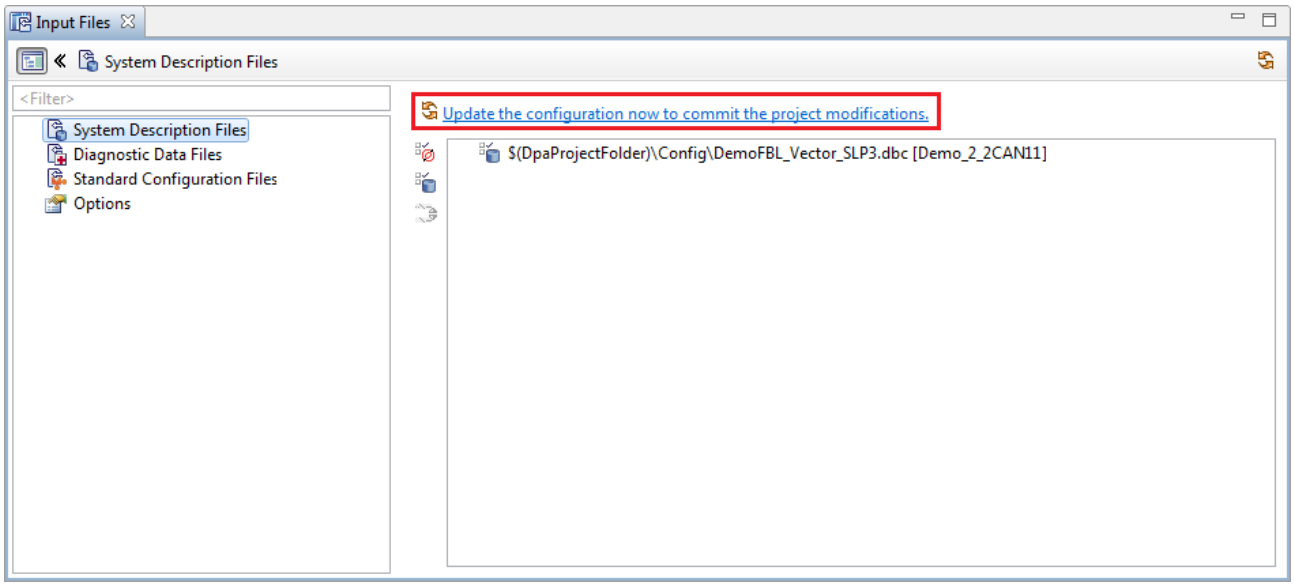

Figure 4-1 Input File Configuration

The following steps are used to load network definitions:

1. Load the system description file.
2. If the system description file includes application messages which are not needed, activate the Fbl PDU system description extractor script. This script removes the unneeded messages from the bootloader configuration (see Figure 4-2 VASE Script Confguration). The script can be found in the directory .\ DaVinciConfigurator\VASE-Scripts\FblPduSystemDescrExtractor of your delivery.
3. After network description and VASE script have been configured, use the link “Update the configuration now to commit the project modifications” to load the file into the project. The link is marked red in Figure 4-1 Input File Configuration.

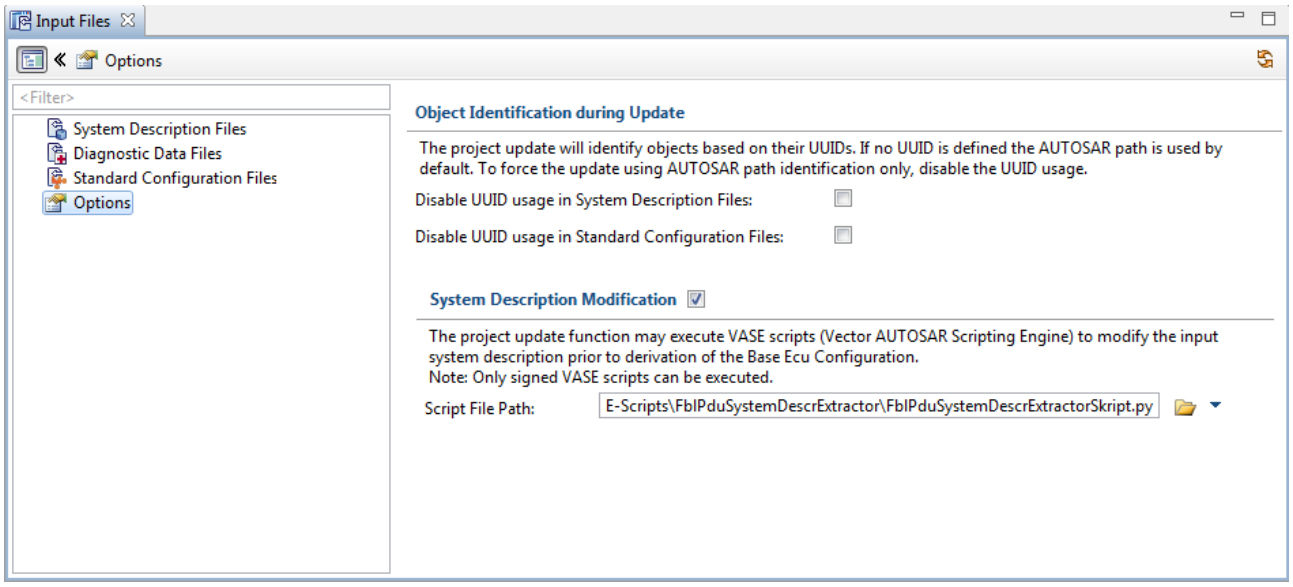

Figure 4-2 VASE Script Confguration

4.2.1.2 Component Selection

Most of the components needed to configure the bootloader are selected automatically, but it is necessary to add some components manually and remove other, automatically added components.

The component selection is done in the project settings tab of the DaVinci Configurator project:

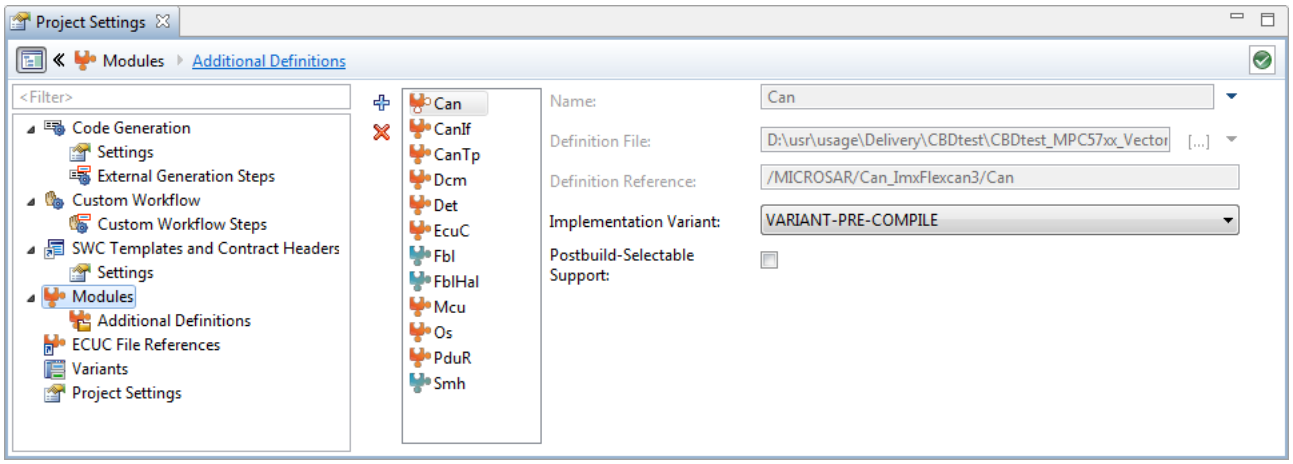

Figure 4-3 Module Selection in Project Settings

Depending on your setup, the following components must be selected (grey: automatically selected and part of SIP, red: manually added and part of SIP, blue: part of AUTOSAR standard definition):

Configuration / Component Setup	CAN Bootloader	Option: CAN NM	FlexRay Bootloader	Option: FlexRay NM	Eth FBL SoAd	Eth FBL DoIP	Option: Ethernet NM
Can	■						
CanIf	■						
CanNm		■					
CanTp	■						
CanSM		■					
ComM		■	■		■	■	
Dcm	■		■		■	■	
Det	■		■		■	■	
DoIP						■	
EcuC	■		■		■	■	
Eth					■	■	
EthIf					■	■	
EthNm							■
EthSM					■	■	
EthTrcv					■	■	
Fbl	■		■		■	■	
FblHal	■		■		■	■	
Fr			■				
FrIf			■				
FrNm				■			
FrSM			■				
FrTp			■				
FrTrcv			■				
Mcu	■		■		■	■	
Nm		■		■			■
Os	■						
PduR	■		■		■	■	
Smh	■		■		■	■	
SoAd					■	■	
TcpIp					■	■	

Table 4-1 Bootloader Component Selection

Note
Different variants (bus systems and options) can be combined. In this case, the modules required for a configuration have to be merged into one module set.

Caution
If a new project is created, unneeded modules like EcuM can be selected by default. These modules have to be removed from the module configuration.

4.2.2 DCM Module Configuration

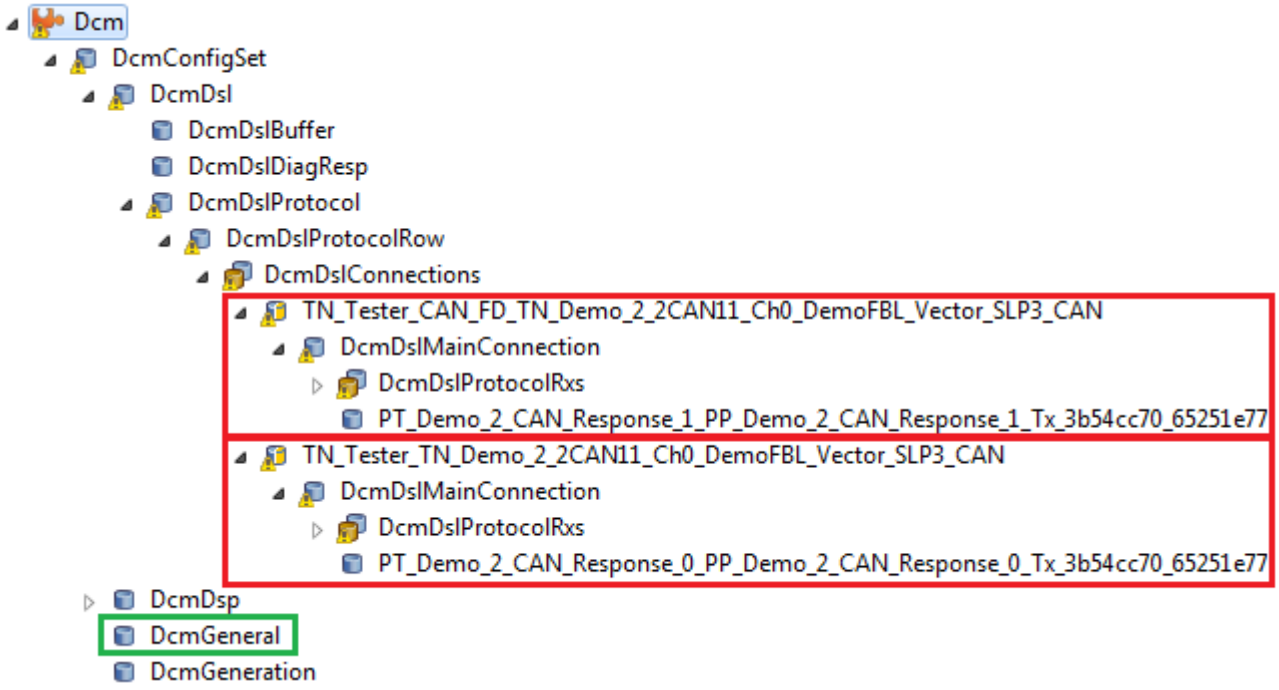

Figure 4-4 Overall DCM Module Structure

The DCM module is used to configure the diagnostic connections of the ECU. The connections are normally set automatically by DaVinci Configurator based on the connections defined in the project's system description file.

Note
The module includes some settings which will be used in future versions of the module but are not used yet. The important sections are marked red and green in Figure 4-4.

4.2.2.1 DcmDsl Section

The DcmDsl section includes settings which are connection-related:

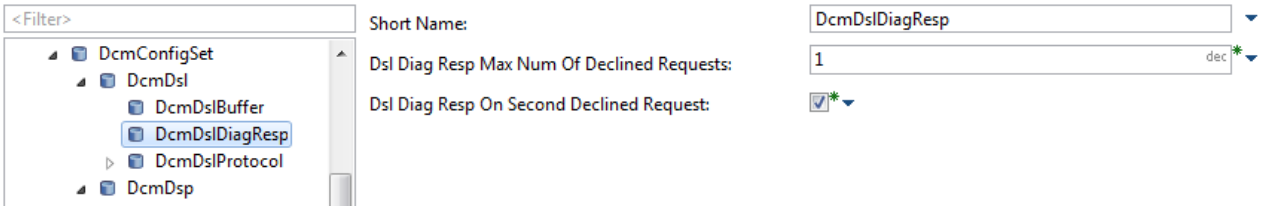

Figure 4-5 Configuration of Diagnostic Responses

On systems with more than one diagnostic connections, only diagnostic request can be handled at a time. This section permits to configure the handling of additional requests if a request is processed by the main connection.

Setting	Description
Dsl Diag Resp Max Num Of Declined Requests	Configure how many diagnostic connections can be used to decline requests. Should be configured to the amount of diagnostic connections – 1.
Dsl Diag Resp On Second Declined Request	If this switch is enabled, and if further requests are received on other connections, a negative response is sent for these requests. If this option is disabled, further requests are ignored.

Table 4-2 DcmDsl Section

4.2.2.2 DcmGeneral Section

The DcmGeneral section includes settings which are not connection-related:

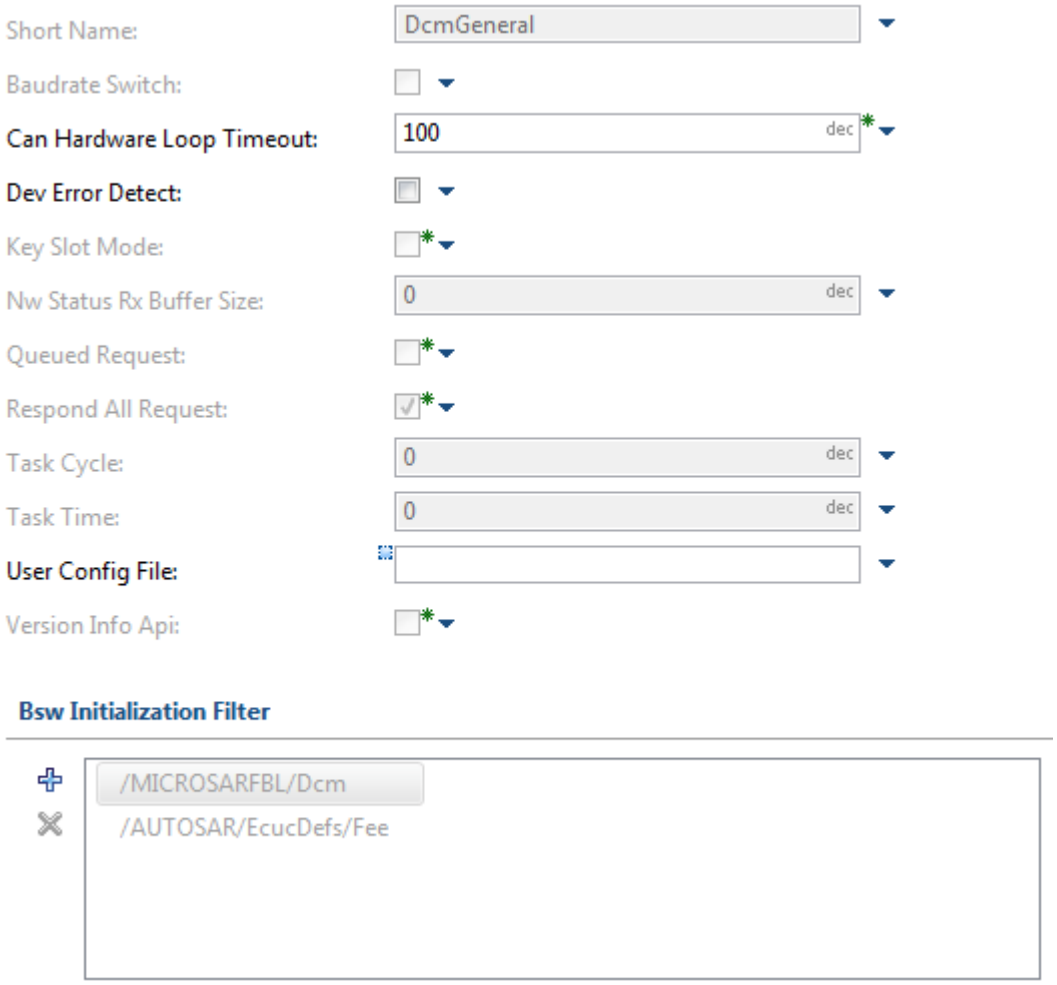

Figure 4-6 DcmGeneral Section

Setting	Description
Baudrate Switch	Activate interface for bus driver baud rate switch. Currently available for CAN only (OEM dependent).
Can Hardware Loop Timeout	Timeout value for CAN hardware loops (in cycles, typically 1 ms).
Dev Error Detect	Switches the Default Error Tracer (Det) detection and notification ON or OFF.
Key Slot Mode	Enable switching from key slot to all slot mode (FlexRay only for certain OEMs).
Nw Status Rx Buffer Size	Buffer size for the network status reception buffer.
Queued Request	Queued Request Feature (certain OEMs only).
Respond All Request	Currently not used.
Task Cycle	Currently not used.
Task Time	Currently not used
User Config File	Path to an external user configuration file used by the Configurator 5 generation tool.
Version Info Api	Currently not used.
BSW Initialization Filter	Autosar modules which shouldn't be initialized automatically by the FblCw component can be added here.

Table 4-3 DcmGeneral Section

4.2.2.3 DcmDslConnections

The DcmDslConnections category includes one container for each diagnostic connection. Physical requests, functional requests and responses are mapped to this connection. The Pdus of the underlying TPs are mapped here.

There are settings for each connection which can be configured:

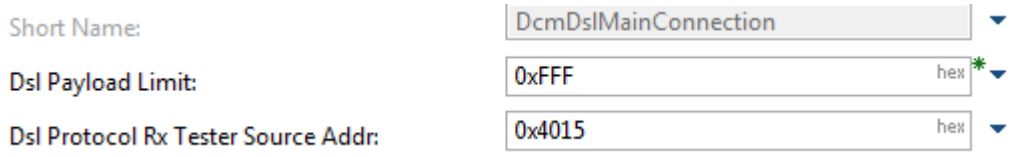

Figure 4-7 Connection Configuration

Setting	Description
Dsl Payload Limit	Payload limitation for each connection. Can be used to handle different payload sizes for each connection.
Dsl Protocol Rx Tester Source Addr	Tester source address used to identify the diagnostic connection. This value must be set to the same value in application software and bootloader configuration.

Table 4-4 Connection Configuration

4.2.3 FlexRay Interface Module Configuration

In addition to the standard FrIf configuration, a user configuration file must be configured in bootloader context.

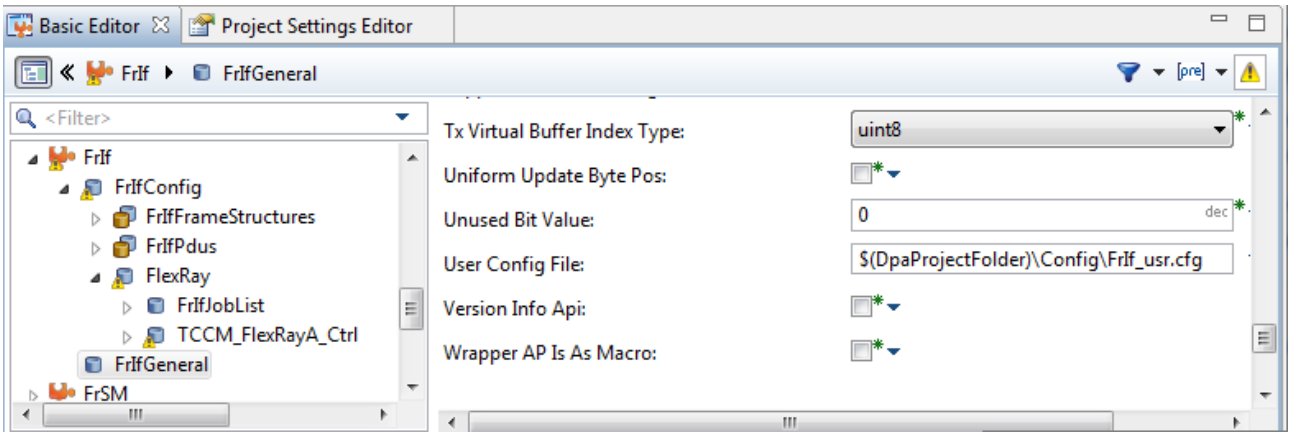

Figure 4-8 FrIf User Configuration File

FrIf user configuration file

The following define must be set with the user configuration file:

```
#define FRIF_CUSTOM_JLE_SCHEDULING STD_ON
```

4.3 Manual Configurations

4.3.1 ComM Channel Configuration

If diagnostic connections use more than one ComM channel, the diagnostic connections must be assigned to a ComM channel. This assignment currently must be done using a user configuration file:

Example

Assign diagnostic connections to ComM channels:

```
#define FBL_CW_DCM_COMM_MAPPING { ComMConf_ComMChannel_0, ComMConf_ComMChannel_1}
```

4.4 Compiler and Linker Settings

To sustain communication during non-volatile memory operations, e.g. sending RCR-RP messages during flash programming, either parts of the communication stack or the complete bootloader have to be linked to RAM. Please make sure you adapt your linker

control file and startup code to generate an image in flash memory and copy this image to RAM during startup.

Example
Please have a look at the linker control file and the MemMap.h file of the FBL demonstration project of your delivery for an example.

Changes
With the use of the MICROSAR communication stack this requirement isn’t limited to the pipelined programming use-case.

5 Interaction with Application Software

The transition between bootloader and application software is often done by diagnostic services, e.g. a Session Control – Programming Session or a ECU Reset – Hard Reset service. These services can be received by bootloader or application, but the response is sent by the other part of the ECU software. In this case, the diagnostic connection which has been used to receive the request must be used to send the response.

If more than one diagnostic connection is configured, the information which diagnostic connection is used must be shared between bootloader and application. Additionally, an interface which ensures that the connections can be identified on both sides is needed because the connections used by bootloader and application are not necessarily the same.

The bootloader provides two callback functions to read the connection information from the application and to store it to be used by the application. Please see the functions ApplFblReadDcmDslRxTesterSourceAddr and

ApplFblWriteDcmDslRxTesterSourceAddr for details.

Note
The transition between bootloader and application is normally done by a reset of the ECU. The data which is handed over between bootloader and application must be stored in a memory area which isn't affected by a reset, e.g. uninitialized RAM or non-volatile memory.

5.1 Address Configuration

The tester source address must be configured to the same value in application and bootloader. The application configuration is shown in Figure 5-1, the bootloader configuration is shown in Figure 4-7.

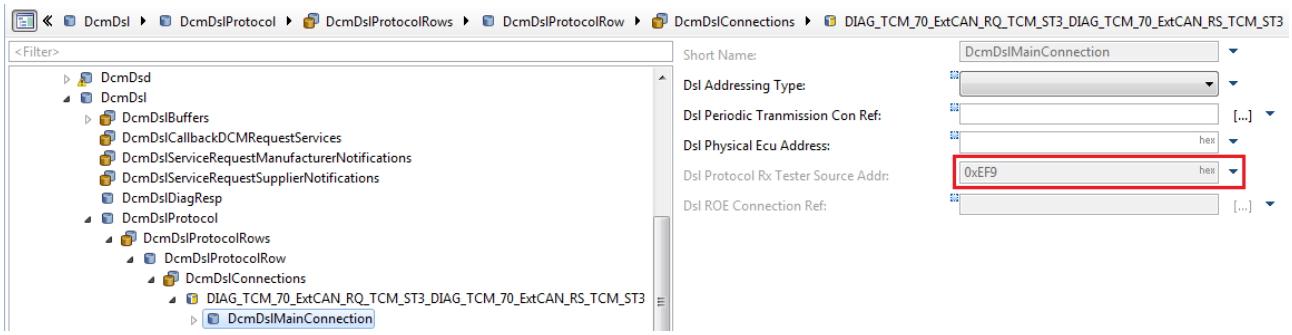

Figure 5-1 Dsl Protocol Rx Tester Source Addr in MICROSAR project

5.2 Transition from Application to Bootloader

When running in the application, the bootloader is started with a diagnostic service and the response to this service is sent by the bootloader, the connection address must be handed over to the bootloader. This is normally done during the service Session Control – Programming Session.

The function Dcm_SetProgConditions(Dcm_ProgConditionsPtrType progConditions) in the application software (MSR) package can be used to activate the FBL. The following data must be handed over to the bootloader:

Data Element	Source
Programming Request Flag	Constant: kEepFblReprogram (0xB5u)
Tester Source Address	progConditions->TesterSourceAddr

Table 5-1 Transition to Bootloader

The data must be set in case the function is called after a Session Control – Programming Session service has been received. The application software should send a RCR-RP message and issue a reset. After this reset, the bootloader should send the response message.

5.3 Transition from Bootloader to Application

If the bootloader is left with a diagnostic service (e.g. Session Control – Default Session or ECU Reset – Hard Reset) and the feature “Response After Reset” is activated, the application software must send the response to the request which has been received by the bootloader.

The function Dcm_GetProgConditions(Dcm_ProgConditionsPtrType progConditions) is used to send the response message.

Data Element	Source
Function return value	Constant: DCM_WARM_START
progConditions->Sid	Reset Response Flag: > RESET_RESPONSE_SDS_REQUIRED: 0x10 > RESET_RESPONSE_ECURESET_REQUIRED: 0x11 > RESET_RESPONSE_KEYOFFON_REQUIRED: 0x11
progConditions->SubFuncId	Reset Response Flag: > RESET_RESPONSE_SDS_REQUIRED: 0x01 > RESET_RESPONSE_ECURESET_REQUIRED: 0x01 > RESET_RESPONSE_KEYOFFON_REQUIRED: 0x03
progConditions->TesterSourceAddr	Tester Source Address

Table 5-2 Transition from Bootloader

6 API Description

Only the API of user-adaptable callback functions is listed here. The bootloader internal APIS to the communication stack and the diagnostics are not included.

6.1 Callback-Function API

6.1.1 Overview

Function API	Typical location
tFblResult ApplFblReadDcmDslRxTesterSourceAddr(vuint8 * buffer)	fbl_ap.c
tFblResult ApplFblWriteDcmDslRxTesterSourceAddr(vuint8 * buffer)	fbl_ap.c
void ApplFblCanBusOff(void)	fbl_ap.c
vuint8 ApplFblCwWakeUp(void)	fbl_ap.c

Table 6-1 Callback-Function API

6.1.2 Function Description

6.1.2.1 ApplFblReadDcmDslRxTesterSourceAddr

Prototype	
tFblResult	ApplFblReadDcmDslRxTesterSourceAddr(vuint8 * buffer)
Parameter	
`buffer`	Pointer to target address buffer
Return code	
kFblOk kFblFailed	Read operation successful Read operation failed
Functional Description	
This function is called to initialize the active connection from NV-memory. The connection is identified by the tester source address parameter of each diagnostic connection.	
Particularities and Limitations	
> This function is called when the FBL is started after reset.	

Table 6-2 FpplFblReadDcmDslRxTesterSourceAddr

6.1.2.2 ApplFblWriteDcmDslRxTesterSourceAddr

Prototype	
tFblResult	ApplFblWriteDcmDslRxTesterSourceAddr(vuint8* buffer)
Parameter	
`buffer`	Pointer to target address buffer
Return code	
kFblOk kFblFailed	Write operation successful Write operation failed
Functional Description	
This function is called to store the active connection in NV-memory. The connection is identified by the tester source address parameter of each diagnostic connection.	

Table 6-3 ApplFblWriteDcDslRxTesterSourceAddr

6.1.2.3 ApplFblCanBusOff

Prototype	
void	ApplFblCanBusOff(void)
Parameter	
–	-
Return code	
-	-
Functional Description	
This call-back function is called if a CAN controller enters bus-off state.	
Particularities and Limitations	
> Only applies to CAN connections. > Currently, no controller dependent handling is implemented. The default bus-off call-back is used all the time.	

Table 6-4 ApplFblCanBusOff

6.1.2.4 ApplFblCwWakeUp

Prototype	
vuint8	ApplFblCwWakeUp(void)
Parameter	
–	-
Return code	
kFblOk kFblFailed	Wake-up detected Otherwise
Functional Description	
This call-back function is called if a wake-up condition is detected during sleep mode.	

Table 6-5 ApplFblCwWakeUp

7 Glossary and Abbreviations

7.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
CAN	Controller Area Network
ECU	Electronic Control Unit
FBL	Flash Bootloader
FR	FlexRay
ISO	International Organiziation for Standardization
MSR	MICROSAR
PDUR	PDU Router
RCR-RP	Response Correctly Received, Response Pending
TP	Transport Protocol
VASE	Vector AUTOSAR Scripting Engine

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com