

dSPACE Calibration and Bypassing Service

Implementation

For Service Version 2.4

Release 2021-A – May 2021

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: <http://www.dspace.com/go/locations>
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: <http://www.dspace.com/go/supportrequest>. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit <http://www.dspace.com/go/patches> for software updates and patches.

Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2005 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

About This Reference	7
License Agreement for the dSPACE Calibration and Bypassing Service	9
License Agreement.....	9
Introduction to the dSPACE Calibration and Bypassing Service Implementation	13
Basics on the dSPACE Calibration and Bypassing Service.....	14
System Components.....	15
Software Layers.....	20
dSPACE Calibration and Bypassing Service Files.....	22
Elementary Data Types.....	24
Integration of the dSPACE Calibration and Bypassing Service	25
dSPACE Calibration and Bypassing Service API.....	25
Integration of the dSPACE Calibration and Bypassing Service in the ECU Code.....	27
Basics on Service Integration with Multicore ECUs.....	31
Data Structures Used by the dSPACE Calibration and Bypassing Service	33
Basics on the Data Structures.....	34
Service Configuration Section (SCS).....	37
Table Pointer Section (TPS).....	37
Address Tables (AT).....	38
Extended Table Pointer Section (ETPS).....	39
Service Pointer Table (SPT).....	39
dSPACE Calibration and Bypassing Service Mechanisms	41
Extended Bypassing Mechanisms.....	41

Alive and Version Information Mechanism.....	47
Subinterrupt Handling Mechanism.....	55
Block Copy Mechanism.....	58
ECU Flash Programming Mechanism.....	59
dSPACE Calibration and Bypassing Service API Functions	63
dsecu_service_init.....	64
dsecu_service.....	64
dsecu_service_state.....	66
dsecu_service_background.....	67
dsecu_service_buffer_sync.....	68
dsecuboot_check.....	69
dSPACE Calibration and Bypassing Service Custom API Functions	71
Basics on the dSPACE Calibration and Bypassing Service Custom API.....	72
dsecu_custom_flash_program_start.....	73
dsecu_custom_flash_erase_range.....	74
dsecu_custom_flash_program_range.....	75
dsecu_custom_flash_decode.....	76
dsecu_custom_flash_finish.....	77
dsecu_custom_command.....	78
dsecu_custom_copy.....	79
dsecu_custom_cal_page_switch.....	80
dsecu_custom_cal_absolute_access.....	81
dsecu_custom_cal_paged_access.....	82
dsecu_custom_cal_paged_access_finished.....	83
dsecu_custom_trigger_hw_int.....	84
dsecu_custom_timeout_timestamp_get.....	85
dsecu_custom_timeout_pending_check.....	86
dSPACE Calibration and Bypassing Service Configuration Options	87
Basics on Configuring Features of the dSPACE Calibration and Bypassing Service.....	88
Basics on Configuration Options.....	88
Configuration of the Enabling/Disabling Features.....	90
DSECU_SERVICE_REMOVED.....	90
DSECU_DAQ_BYPASS_SERVICE.....	91

DSECU_BYPASS.....	92
DSECU_CALIBRATION_COMMANDS.....	92
DSECU_CUSTOM_COMMANDS.....	93
DSECU_FLASH_COMMANDS.....	94
Configuration of General Features.....	96
DSECU_NUMBER_TOOLS.....	96
DSECU_INT_STATUS_VAR_DECL.....	97
DSECU_INT_SAVE_AND_DISABLE.....	98
DSECU_INT_RESTORE.....	99
DSECU_POINTER_QUALIFIER.....	100
DSECU_INLINE.....	101
DSECU_DPMEM_OFFSETADDRx.....	101
DSECU_DPMEM_SIZEx.....	102
DSECU_POD_TYPEx.....	103
DSECU_PROCESSOR_TYPE.....	104
DSECU_BYTE_WRITE_SUPPORTED.....	105
Configuration of Calibration Features.....	106
DSECU_CAL_NUMBER_OF_PAGES.....	106
DSECU_CAL_PAGED_ADDRESSING.....	107
DSECU_CAL_ABSOLUTE_ADDRESSING.....	108
DSECU_CAL_MEMORY_COPY_METHOD.....	108
Configuration of DAQ and Bypassing Features.....	110
DSECU_SCS_LOCATION.....	111
DSECU_SCSOFFSETADDRx.....	112
MORE_ATs_FOR_ONE_SRV.....	112
DSECU_DOUBLE_BUFFER.....	113
DSECU_BLOCK_COPY.....	114
DSECU_xxBIT_SUPPORT.....	115
DSECU_BITS_SUPPORT.....	116
DSECU_ECU_WAITS.....	117
DSECU_FAILURE_CHECKING.....	118
DSECU_TIMEOUT_CUSTOM_FUNCTION.....	119
DSECU_TIMEOUT_TIMESTAMP_TYPE.....	121
DSECU_TIMEOUT_SCALING_10US.....	122
DSECU_FAILSAFE_PAGE.....	123
DSECU_CUSTOM_HW_TRIGGER.....	124
DSECU_SUBINT_TYPE.....	125
DSECU_SINT_NUMBER.....	126
DSECU_MAX_SERVICE_COUNT.....	127

Configuration of ECU Flash Programming Features.....	128
Overview of Options for Configuring ECU Flash Programming	
Features.....	129
DSECU_MAILBOX_ADDRx.....	129
DSECU_FLASH_MAX_SECTORS.....	130
DSECU_FLASH_SECTOR_LIST.....	131
DSECU_FLASH_MAX_CODECHECKS.....	132
DSECU_FLASH_CODECHECK_LIST.....	133
DSECU_APPLICATION_LOCK_INIT.....	134
DSECU_APPLICATION_LOCK_LOOP.....	135
DSECU_APPLICATION_VALID_CHECK.....	135
DSECU_APPLICATION_VALID_ID_ADDRESS.....	136
DSECU_APPLICATION_VALID_ID_VALUE.....	137
 dSPACE Calibration and Bypassing Service Interface	
Description.....	139
Interface Description.....	139
 Index.....	141

About This Reference

Contents

This document gives you information on how to implement the dSPACE Calibration and Bypassing Service in your ECU code. The document also provides detailed information on the available functions and macros needed to specify and configure the service.

Note






The *dSPACE Calibration and Bypassing Service* is part of the installation of ECU Interface Software.



After you install and decrypt ECU Interface Software, you will find the `dSPACECalibrationAndBypassingService_<version>.exe` file in the `%ProgramData%\dSPACE\<InstallationGUID%\dsECU\Services` folder. Run it to install the service and its documentation in a folder of your choice.

You can access the `%ProgramData%\dSPACE\<InstallationGUID>` folder via a shortcut in the Windows Start menu below dSPACE RCP and HIL <version>.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
 DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
 WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
 CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
 NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
 Note	Indicates important information that you should take into account to avoid malfunctions.

Symbol	Description
Tip	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>

or

%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>

Documents folder A standard folder for user-specific documents.

%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>

Accessing dSPACE Help and PDF Files


After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.

License Agreement for the dSPACE Calibration and Bypassing Service

Introduction

If you want to work with the dSPACE Calibration and Bypassing Service, you have to accept the License Agreement first.

License Agreement

License agreement

Note

*IMPORTANT – USE OF THIS SERVICE IS SUBJECT TO LICENSE RESTRICTIONS
READ THIS LICENSE AGREEMENT CAREFULLY BEFORE USING THE SERVICE*

This license is a legal Agreement between you, the end user, either individually or as an authorized representative of the company acquiring the license, and dSPACE GmbH acting directly or through its local dSPACE companies or authorized distributors (collectively “dSPACE”), concerning the use of the C code containing the dSPACE Calibration and Bypassing Service (hereinafter referred to as “the Service”) together with any other materials which are provided for use in connection with the Service, including without limitation the executable for installation of the Service, any associated user manual and internal documentation (hereinafter collectively referred to as “the Program”).

BY IMPLEMENTING THE EXECUTABLE AND INSTALLING THE PROGRAM, YOU AGREE TO COMPLY WITH THE FOLLOWING TERMS AND RESTRICTIONS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT INSTALL OR USE THE PROGRAM AND PROMPTLY RETURN IT TO THE PLACE WHERE YOU OBTAINED IT, OR DELETE THE PROGRAM IF YOU RECEIVED IT ELECTRONICALLY.

1. Grant of License

Unless explicitly agreed otherwise, dSPACE grants you a nonexclusive license to use the Program and execute the Service as described in the respective product description or documentation for the sole purpose of product development involving dSPACE tools.

2. Restrictions of Use

You may not market, distribute or transfer copies of the Program, in whole or in part, to third parties (including any subsidiary, affiliate or company under common control with you) or transfer the Program (directly or indirectly) via Internet or network applications (such as Citrix, Microsoft Remote Desktop or other terminal servers) or grant third parties any access to the Program by any means. You may not rent, lease or loan the Program.

These restrictions do not prevent you from providing compiled object code versions of the Service as part of your own ECU code to third parties, subject to the condition that:

- This takes place in the course of a project where (amongst others) dSPACE tools are used, *and*
- The code is used for the sole purpose of product development and not for use in any end product or production.

The recipient of your respective ECU code needs to be instructed accordingly and shall undertake to comply with these restrictions and to agree to the Limitation of Liability according to Clause 4 hereunder. dSPACE reserves the right to ask for written confirmation that appropriate instructions have been issued.

Upon request and at the sole discretion of dSPACE, you may be granted permission to provide the Service itself, in whole or in part, to third parties as part of your own ECU source code, subject to the conditions stated above. To be valid, such permission needs to be granted in writing by dSPACE.

For the avoidance of doubt, in any case any transfer of or granting of access to parts of the Program other than the Service itself is explicitly prohibited.

3. Confidentiality

dSPACE considers the Program to contain valuable intellectual property of dSPACE, the unauthorized disclosure of which could cause irreparable harm to dSPACE. You agree to use reasonable efforts not to disclose the Program to any third parties (including any subsidiary, affiliate or company under common control with you) and not to use the Program other than for the purposes authorized by dSPACE.

4. Limitation of Liability

The Program was designed and tested solely for use in research and product development and is supplied to you by dSPACE exclusively for this purpose. It must be put into operation exclusively by suitably trained and expert operating personnel under strict compliance with the safety measures described in the software documentation. Any use of the Program or compiled object code versions of the Service for purposes and under conditions other than the above, including but not only any use in end products, constitutes inappropriate use.

Any liability by dSPACE under mandatory law, including but not restricted to product liability law, for damages of any kind that may be caused by using the Program or compiled object code versions of the Service in areas other than product development shall be limited, even to the point of total exclusion, as the case may be. In the event of claims by third parties against dSPACE that are due to such inappropriate use of the Program or of compiled object code versions of the Service by you or with your permission, you agree to indemnify dSPACE against all such claims.

In addition, the regulations on liability according to the General Terms and Conditions of dSPACE as attached to any dSPACE offer apply accordingly. A copy of the General Terms and Conditions can also be obtained at info@dspace.de.

5. Miscellaneous

Any amendments or additions to this Agreement must be made in writing and must be expressly marked as such. This also applies to this written form requirement.

In the event that any of the above terms is or becomes invalid, the remaining terms shall continue in full force and effect.

Any failure to enforce, or any waiver of, any right under this Agreement by dSPACE shall not be construed as a waiver of future rights.

The legal regulations shall apply in addition to the terms of this Agreement, except in cases where they conflict with said terms. This Agreement shall be governed by the laws of the Federal Republic of Germany, excluding the UN Convention on Contracts for the International Sale of Goods (CISG).

Paderborn, Germany, is agreed as the exclusive place of jurisdiction for all disputes arising from or in connection with this Agreement, unless a different place of jurisdiction is mandatory on the basis of legal requirements.

Introduction to the dSPACE Calibration and Bypassing Service Implementation

Introduction

The dSPACE Calibration and Bypassing Service is used to control communication between an ECU and a calibration and/or prototyping tool. It can be used for calibration, data acquisition, bypassing and ECU flash programming purposes.

Where to go from here

Information in this section

Basics on the dSPACE Calibration and Bypassing Service.....	14
Gives you an overview of the features of the dSPACE Calibration and Bypassing Service.	
System Components.....	15
Details on the components of a system that uses the dSPACE Calibration and Bypassing Service.	
Software Layers.....	20
The dSPACE Calibration and Bypassing Service has several software layers.	
dSPACE Calibration and Bypassing Service Files.....	22
The dSPACE Calibration and Bypassing Service consists of several header and code files.	
Elementary Data Types.....	24
Overall definitions of data types used by the dSPACE Calibration and Bypassing Service.	

Basics on the dSPACE Calibration and Bypassing Service

Introduction

The dSPACE Calibration and Bypassing Service is used to control communication between an ECU and a calibration and/or prototyping tool. It can be used for calibration, data acquisition, bypassing and ECU flash programming purposes.

Access to ECU application and resources

During development of an ECU it is often necessary to access the application running on the ECU. For example, values that are calculated by the ECU must be measured, or parameters must be calibrated to change the behavior of the ECU control algorithm.

The dSPACE Calibration and Bypassing Service provides access to the ECU application and the ECU resources for calibration, measurement, bypassing and ECU flash programming.

Further requirements

To perform calibration, data acquisition (DAQ), bypassing, or ECU flash programming using the dSPACE Calibration and Bypassing Service, different components are required:

Calibration and DAQ For calibration and DAQ, you need the following items:

- PC with ControlDesk
- ECU with DCI-GSI2
- or*
- RapidPro system used as a stand-alone prototyping ECU
- USB connection between the RapidPro system and the host PC with ControlDesk
- or*
- Ethernet connection between the ECU with DCI-GSI2 and the host PC with ControlDesk

Implementing the dSPACE Calibration and Bypassing Service is necessary in the following cases:

Interface	Calibration	Data Acquisition
DCI-GSI2	✓ / – ¹⁾	✓ / – ²⁾
RapidPro	✓	✓

¹⁾ Depending on the ECU microcontroller and the configured calibration method, calibration might be performed using the dSPACE Calibration and Bypassing Service. In other cases, however, other mechanisms are used on the DCI-GSI2.

Interface	Calibration	Data Acquisition
-----------	-------------	------------------


²⁾ The integration of the dSPACE Calibration and Bypassing Service is optional. If the ECU has a data trace interface, measurement data can be acquired synchronously and consistently without an ECU service.

Bypassing For bypassing, you need the following items:

- PC with RTI Bypass Blockset
- dSPACE prototyping system:
 - Modular system based on DS1006 with DS4121 ECU Interface Board, or
 - MicroAutoBox II with ECU Type 1 board
- ECU with POD
- LVDS connection between the ECU with POD and the dSPACE prototyping system

ECU flash programming For ECU flash programming, you need the following items:

- dSPACE ECU Flash Programming Tool
- DCI-GSI2
- Ethernet connection between the host PC with dSPACE ECU Flash Programming Tool and the DCI-GSI2

Service features	For information on the features provided by the dSPACE Calibration and Bypassing Service, refer to Features of the dSPACE Calibration and Bypassing Service (dSPACE Calibration and Bypassing Service Feature Reference )
------------------	---

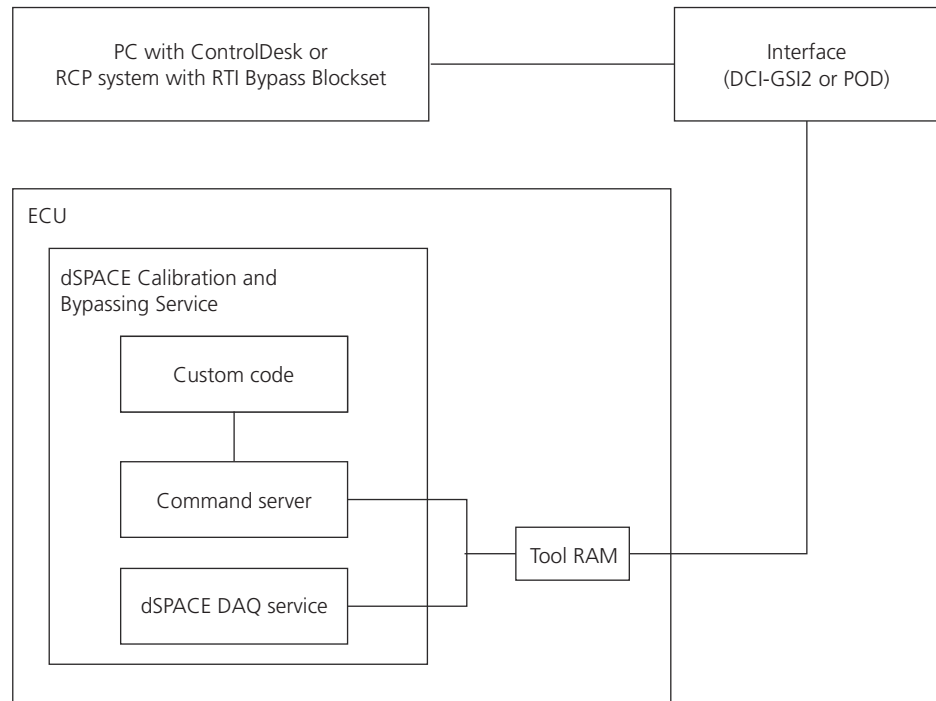
Related topics	<div>Basics</div> <div><div>dSPACE Calibration and Bypassing Service Files.....22</div><div>Elementary Data Types.....24</div><div>Software Layers.....20</div><div>System Components.....15</div></div>
----------------	--

System Components

Introduction	To make use of the dSPACE Calibration and Bypassing Service, your system must have different components.
--------------	--

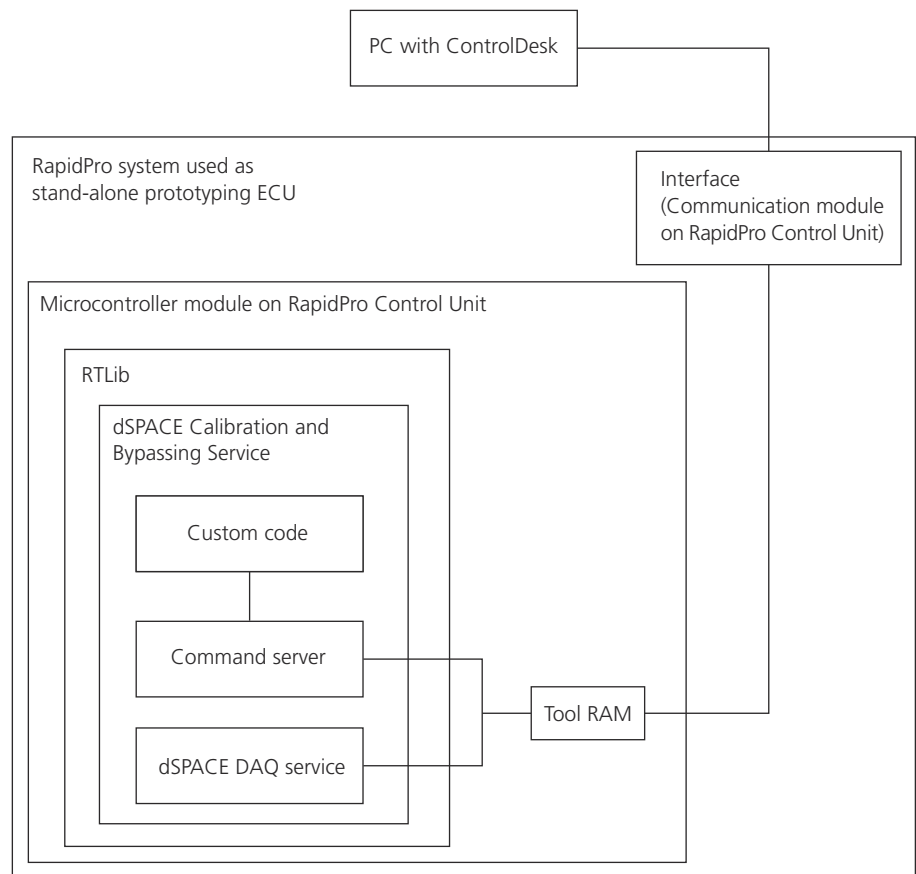
System components for working with an ECU

The following illustration shows the required system components for working with an ECU:



System components for working with a RapidPro system

The following illustration shows the required system components for working with a RapidPro system used as a stand-alone prototyping ECU:



Host applications: ControlDesk, RTI Bypass Blockset, dSPACE ECU Flash Programming Tool

A host application is required to access the ECU or RapidPro system via the dSPACE Calibration and Bypassing Service.

Host application for calibration and data acquisition You can use ControlDesk on your PC as a host application to calibrate parameters and perform data acquisition (DAQ) on an ECU or RapidPro system used as a stand-alone prototyping ECU.

Host application for ECU function bypassing You can use the RTI Bypass Blockset to implement communication between an ECU and the dSPACE prototyping system for ECU function bypassing.

Note

Function bypassing is currently not supported for RapidPro systems used as stand-alone prototyping ECUs.

Host application for ECU flash programming A flash tool, for example, the dSPACE ECU Flash Programming Tool, can access the dSPACE Calibration and Bypassing Service to program the ECU's flash memory.

Note

For RapidPro systems used as stand-alone prototyping ECUs, building and downloading real-time applications to the RapidPro hardware is a process which must be started manually.

Host interface

The connection to the host interface is established via

- The Universal Serial Bus (USB) for calibration, data acquisition, or ECU flash programming purposes
- Ethernet for calibration, data acquisition, or ECU flash programming purposes
- The low-voltage differential signaling (LVDS) interface for bypassing purposes

Note

Currently, only the PODs have an LVDS interface which can be used for bypassing.

Host interface for calibration and data acquisition When the dSPACE Calibration and Bypassing Service is used for calibration or data acquisition, the commands and data sent by the host PC are not received directly by the ECU, but by the host interface. The host interface forwards them to the ECU. This is realized via data structures located in the tool RAM. The location of the tool RAM depends on the connected hardware:

- With the DCI-GSI2, the tool RAM is in the ECU RAM.
- With a DPMEM-based POD, the tool RAM is in a dual-port RAM.
- With the communication module (COM module) of the RapidPro system used as a stand-alone prototyping ECU, the tool RAM is in a dual-port RAM.

The data structures are accessed either by the host interface or by the ECU.

Hence, the host interface is a piece of hardware that implements the host connection itself and a dual-port RAM (for a DPMEM-based POD or the COM module of the prototyping ECU) or a debug interface to access the data structures in the ECU's RAM (for the DCI-GSI2).

The software running on the host interface contains, among other modules, a host interface module and the master of the dSPACE Calibration and Bypassing Service. The host interface module receives the commands from the host application and calls corresponding functions in the dSPACE Calibration and Bypassing Service master. The master is responsible for setting up the tables for DAQ, and for calling commands on the ECU.

Host interface for bypassing When the dSPACE Calibration and Bypassing Service is used for bypassing, the software that controls the service runs on the prototyping system. That means that the master of the dSPACE Calibration and Bypassing Service is on the prototyping system. The host interface is used to

transfer the data sent by the bypassing system to the ECU. It simply forwards the read/write commands.

ECU software

The dSPACE Calibration and Bypassing Service consists of several modules:

- dSPACE DAQ service module that is needed to perform data acquisition and/or bypassing
- Command Server module that processes the commands sent by the host interface

The Command Server module receives the commands from the host interface, processes them, and sends back their return values.

- Customer-specific module that implements the flashing algorithm and additional commands

Some functions must be adapted to a specific ECU. For example, flashing algorithms depend on the ECU's flash memory. In addition, there is a custom command interpreter which can be used to add customer-specific commands. The custom command feature is available only as part of a dSPACE engineering service, because it requires modifications to the DCI-GSI2 firmware.

For a RapidPro system used as a stand-alone prototyping ECU, the custom layer is already implemented for the given processor module. Hence, the appropriate functions are integrated in the dSPACE Calibration and Bypassing Service, which is included in the RTLib (real-time library).

Service configuration

The dSPACE Calibration and Bypassing Service is widely configurable and can be adapted to the needs of a specific project. For example, it is possible to use only the DAQ Service module. In that case, no commands can be sent to the ECU, which means that features like ECU flash programming are not available.

For a RapidPro system used as a stand-alone prototyping ECU, the service configuration is fixed. It contains all the commands required to use the features mentioned above.

Related topics

Basics	
Basics on the dSPACE Calibration and Bypassing Service.....	14
dSPACE Calibration and Bypassing Service Files.....	22
Elementary Data Types.....	24
Software Layers.....	20

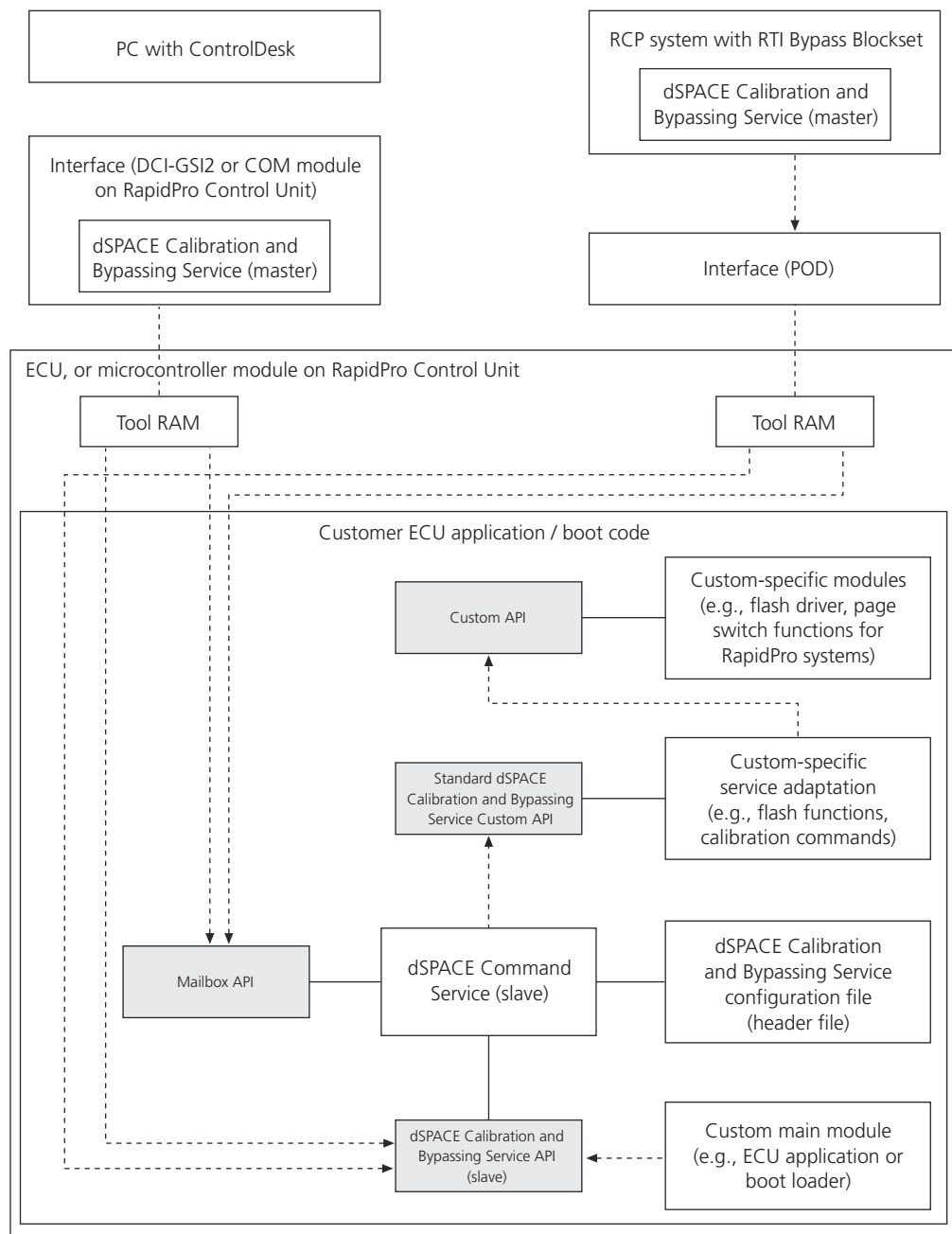
Software Layers

Software layers and APIs

Several software components and APIs are involved in using the dSPACE Calibration and Bypassing Service. They are included in the ECU application or ECU boot code with the following modules integrated:

- Standard dSPACE Command Service, which provides the Mailbox API used by the dSPACE Calibration and Bypassing Service host. The dSPACE Calibration and Bypassing Service is configured to meet your ECU requirements through a separate header file. It also provides an easy-to-use Service (slave) API to integrate the service into your ECU application.
- Standard customer flashing module (custom-specific API).
- Adaptation layer which maps the standard dSPACE Flash API to the custom-specific functions (see [dSPACE Calibration and Bypassing Service Custom API Functions](#) on page 71). The module is optional, but it must be present for ECU flash programming support. The flashing functions are not provided by dSPACE, but in most cases by the flash supplier.

The following illustration shows the different software layers and APIs and their dependencies:

**Note**

The following elements must be adapted to specific requirements:

- dSPACE Calibration and Bypassing Service configuration file (header file)
- Custom main module
- Custom-specific modules

Related topics**Basics**

Basics on the dSPACE Calibration and Bypassing Service.....	14
dSPACE Calibration and Bypassing Service Files.....	22
Elementary Data Types.....	24
System Components.....	15

dSPACE Calibration and Bypassing Service Files

Introduction

The functions and configuration options used to implement the dSPACE Calibration and Bypassing Service are contained in different header and code files which you must compile and link to the ECU code.

File structure

The following tables show the header and code files for the dSPACE Calibration and Bypassing Service. Different files are required, depending on the task you want to perform.

Performing DAQ and bypassing To perform data acquisition or bypassing, the following header and code files are required:

File Type	File Name	Description
Common files ¹⁾	dsECUcmd.c	Command server
	dsECUcmd.h	Include file for the command server
	dsECUcustom.h	Prototypes for commands
	dsECUsvc.c	DAQ and bypassing
	dsECUsvc.h	Include file for DAQ and bypassing
	dsECUcmd_include.h	Include file shared between the service slave and master
Custom files ²⁾	dsECUcfg.h	Configuration file for the complete service
	dsECUcustom.c	File for custom functions and commands

¹⁾ The common files are fixed files containing the dSPACE Calibration and Bypassing Service and therefore must not be altered.

²⁾ The custom files are used for custom configuration and adaptation. In the case of a RapidPro system used as a stand-alone prototyping ECU, the custom files are fixed files. They are integrated in the dSPACE Calibration and Bypassing Service, which is included in the RTLib.

Note

You must compile and link the files to your ECU application.

Building a custom flash kernel The flash functions of the dSPACE Calibration and Bypassing Service are not integrated into the standard ECU

application, but into a separate flash kernel. This means that the functionalities of the service files listed in the table above must be integrated into the flash kernel for ECU flash programming purposes.

Note

A flash kernel has to be adapted to the specific requirements of an ECU and its software. Contact dSPACE to discuss the next steps for creating a flash kernel for your ECU.

Performing ECU flash programming To perform ECU flash programming, the flash kernel has to be activated (regardless of whether you use a flash kernel file provided by dSPACE or a self-programmed flash kernel). The **dsECUboot** module must be included in the ECU code for this purpose.

The **dsECUboot** module contains the following header and code files:

File Type	File Name	Description
Common files ¹⁾	dsECUboot.c	Boot check function
	dsECUboot.h	Include file for the boot check function
Custom files ²⁾	dsECUbootcfg.h	Specific configuration file for the dsECUboot module

¹⁾ The common files are fixed files containing the dSPACE Calibration and Bypassing Service and therefore must not be altered.

²⁾ The custom files are used for custom configuration and adaptation.

Note

- The **dsECUboot** module must only be compiled and linked if you want to use the boot check function.
- The **dsECUboot** module can be integrated into either the ECU application code or the ECU boot code, regardless of the other header and code files which are listed in the first table (see above).

Related topics

Basics

Basics on the dSPACE Calibration and Bypassing Service.....	14
Elementary Data Types.....	24
Software Layers.....	20
System Components.....	15

Elementary Data Types

Data types

The `dsECUcfg.h` and `dsECUbootcfg.h` files define the data types for the dSPACE Calibration and Bypassing Service. The default values listed below match the most ECU processors:

<code>typedef unsigned char</code>	<code>DSECU_UInt8;</code>
<code>typedef signed char</code>	<code>DSECU_Int8;</code>
<code>typedef unsigned short</code>	<code>DSECU_UInt16;</code>
<code>typedef signed short</code>	<code>DSECU_Int16;</code>
<code>typedef unsigned long</code>	<code>DSECU_UInt32;</code>
<code>typedef signed long</code>	<code>DSECU_Int32;</code>
<code>typedef float</code>	<code>DSECU_Flt32;</code>
<code>typedef double</code>	<code>DSECU_Flt64;</code>

Include files

- `dsECUcfg.h`
- `dsECUbootcfg.h`

Related topics

Basics

Basics on the dSPACE Calibration and Bypassing Service.....	14
dSPACE Calibration and Bypassing Service Files.....	22
Software Layers.....	20
System Components.....	15

Integration of the dSPACE Calibration and Bypassing Service

Introduction	The ECU application must be instrumented by the dSPACE Calibration and Bypassing Service.
---------------------	---

Where to go from here

Information in this section

dSPACE Calibration and Bypassing Service API.....	25
The dSPACE Calibration and Bypassing Service provides some API functions for initializing the service and performing command processing.	
Integration of the dSPACE Calibration and Bypassing Service in the ECU Code.....	27
The ECU application must be instrumented by different API functions, depending on the ECU tasks purposes.	
Basics on Service Integration with Multicore ECUs.....	31
Working with ECU microcontrollers that contain more than one processing core might require integrating the dSPACE Calibration and Bypassing Service into multiple cores.	

dSPACE Calibration and Bypassing Service API

Introduction	The dSPACE Calibration and Bypassing Service provides three API functions for initializing the service and performing command processing.
---------------------	---

Service initialization

The service initialization is performed by the `dsecu_service_init` initialization function, which must always be integrated into the ECU code. The function must be called before any other dSPACE Calibration and Bypassing Service function. A typical location in the ECU code is where other initializations of the ECU are performed. You need to access the tool RAM for this, so the chip select must first be suitably programmed.

Foreground service

The foreground service of the dSPACE Calibration and Bypassing Service is called to transfer data to and from the ECU. For data transmission from the ECU to the host, this is done by copying the (measurement) data to the tool RAM, and invoking a trigger on the host interface. When bypassing is performed, data is also copied from the host to the ECU. An interrupt can be used for this.

The foreground service is performed by the `dsecu_service` function, which must only be integrated into the ECU code if data acquisition (DAQ) or bypassing is to be performed. For data acquisition, the function is typically called at the end of an ECU task. The function call can be placed in any function, so that, for example, time-synchronous or crankshaft-synchronous data acquisition can be performed. Two scenarios are possible for bypassing:

- Bypassing using two function calls
One function call is placed at the beginning of the function or ECU task to be bypassed (where the tool reads the arguments of the functions to be bypassed), and one function call is placed at the end of the function or ECU task to be bypassed (where the function results calculated by the bypassing tool are written back to the ECU).
- Bypassing using one function call
The function call is placed at the end of the function to be bypassed, or, if the function results are not used again in the current ECU task, at the end of the task. The tool reads the arguments of the functions to be bypassed, and the function results of the last sampling step calculated by the bypassing tool are written back to the ECU.

Background service

The background service of the dSPACE Calibration and Bypassing Service checks whether the tools are connected to the ECU and whether they are working correctly. This is done by executing an alive check and activating the Command Server. Depending on the configuration of the dSPACE Calibration and Bypassing Service, the background service also processes commands.

The background service is performed by the `dsecu_service_background` function, which must always be integrated into the ECU code. The function must be called repeatedly in the background task of the ECU, or in the task with the largest sample time if no background task is available.

Download and start of the flash kernel

The dSPACE Calibration and Bypassing Service can be used for ECU flash programming with the DCI-GSI2 ECU interfaces. A single boot check function of

the dSPACE Calibration and Bypassing Service must be integrated in your ECU application or the ECU's boot code to allow ECU flash programming.

The flash functions of the dSPACE Calibration and Bypassing Service are not integrated into the standard ECU application, but into a separate flash kernel. When the ECU is to be flashed, the DCI-GSI2 sends a flashing request to the dSPACE mailbox in the tool RAM, and triggers an ECU reset. When the ECU reboots, the boot check function of the dSPACE Calibration and Bypassing Service checks whether there is a flashing request. If there is, the boot check function on the ECU waits until the flash kernel is loaded to the ECU's RAM and started. The ECU then runs from the flash kernel. The ECU is ready to receive flash commands. The flash tool sends a sequence of erase and program commands to flash the new ECU application code or calibration data to your ECU.

The boot check is performed by the `dsecuboot_check` function, which must be called only if ECU flash programming is to be performed with the dSPACE Calibration and Bypassing Service. It is recommended to call the `dsecuboot_check` function as soon as possible in the ECU software (if possible in the ECU's boot code), so that it is possible to flash the ECU even if a part of the ECU software is erroneous or damaged. `dsecuboot_check` must be called before the `dsecu_service_init` function.

Related topics

Basics

[Integration of the dSPACE Calibration and Bypassing Service in the ECU Code..... 27](#)

References

[dSPACE Calibration and Bypassing Service API Functions..... 63](#)

Integration of the dSPACE Calibration and Bypassing Service in the ECU Code

Introduction

The ECU application must be instrumented by dSPACE Calibration and Bypassing Service API functions. Different functions must be called according to the purpose the service is to be used for.

General integration

The initialization function and the background function must always be integrated in the ECU code.

The `dsecu_service_init` initialization function is called during start-up prior to any ECU task.

After execution of all the tasks of the current time raster, the ECU background task is called. The period available for executing the ECU background task

depends on the time remaining in the current time raster. In this part of the ECU code, the `dsecu_service_background` function must be called repeatedly (at least once) until the next time the raster is triggered. The background function checks whether the host interface is still alive and processing commands etc. according to the service configuration.

Note

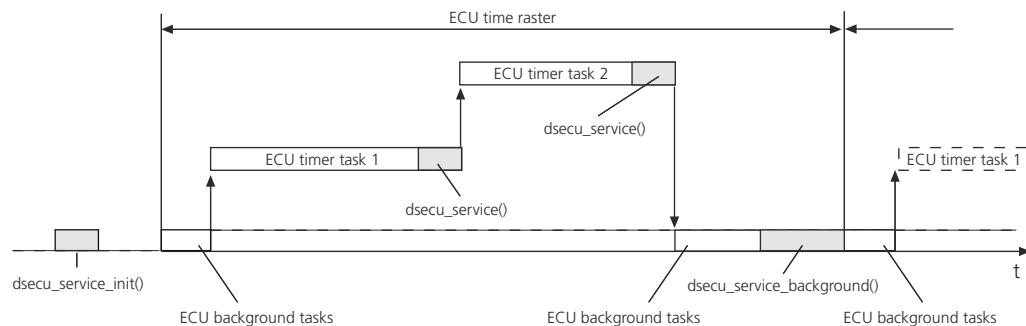
For a RapidPro system used as a stand-alone prototyping ECU, the `dsecu_service_init` and `dsecu_service_background` functions are integrated in the `init()` and `RTLIB_BACKGROUND_SERVICE()` RTLib functions and therefore need not to be called explicitly.

Integration for DAQ and bypassing

The ECU application may consist of several ECU tasks which are called at fixed time intervals (time rasters) or synchronously to specific ECU events (for example, crankshaft-synchronously).

Data acquisition At the end of an ECU task, the `dsecu_service` function is called to trigger data acquisition.

Example The following illustration shows an example of a command flow of the dSPACE Calibration and Bypassing Service within the ECU code for a DAQ scenario:



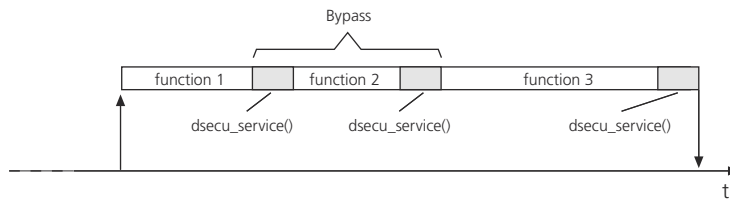
The `dsecu_service_init` initialization function is called during start-up prior to any other ECU task.

After initialization, two tasks are executed successively in the current ECU time raster of the ECU code. At the end of each task, `dsecu_service` is called to perform data transfer.

The ECU background tasks are called by `dsecu_service_background`. This function is to be called at least once until the next time raster is triggered.

Bypassing At the beginning and end of the ECU function or task to be bypassed, the `dsecu_service` function is called to trigger the bypass functionality.

Example To perform bypassing on an ECU with the dSPACE Calibration and Bypassing Service using two foreground service calls, you have to instrument an ECU task as shown by the example in the illustration below.



In this example, the ECU task consists of three functions, and function 2 is to be bypassed. The `dsecu_service` function is called at the beginning of function 2 (to read the function arguments) and at the end of function 2 (to write the calculated data back to the ECU). `dsecu_service` is also called at the end of the ECU task for data acquisition purposes.

Bypassing (cont.) Alternatively, bypassing can be performed using only one `dsecu_service` function call. It is placed at the end of the function to be bypassed, or, if the function results are not used in the further execution of the current task, at the end of the ECU task (see [Foreground service](#) on page 26).

Integration for ECU flash programming

You need to provide specific flash drivers for flashing your ECU flash devices.

Note

The actual flash driver is not part of the dSPACE Calibration and Bypassing Service. You must provide it yourself since it depends on the type of flash devices to be programmed. Many flash device suppliers distribute standard flash drivers for their devices free of charge, for example, as downloads on their Web sites.

To access the flash memory areas, the dSPACE Calibration and Bypassing Service uses a standard dSPACE API which must be adapted to the custom-specific functions.

The following functions must be modified to map the standard dSPACE API to the customer-specific functions:

- `dsecu_custom_flash_program_start()`
- `dsecu_custom_flash_erase_range()`
- `dsecu_custom_flash_program_range()`
- `dsecu_custom_flash_decode()`
- `dsecu_custom_flash_finish()`

For information on the functions, refer to [dSPACE Calibration and Bypassing Service Custom API Functions](#) on page 71.

Integration for calibration

Calibration and page switching depend on your individual solution. You can use pointers to memory areas, each of which represents a calibration page. Calibration is performed by modifying a parameter in the appropriate page, and page switching is done by changing the pointer to the current page. Or you can use the memory overlay units supported by some processor types (for example,

Freescall's MPC565 processor). Page switching can also be performed by reconfiguring the chip select.

The dSPACE Calibration and Bypassing Service provides standard functions for calibration and page switching, contained in a standard dSPACE Calibration and Bypassing Service Calibration API. The implementation of these functions is customer-specific, that is, the API functions must be adapted to the customer-specific functions.

The following functions must be modified to map the standard dSPACE API to the customer-specific functions:

- `dsecu_custom_cal_page_switch()`
- `dsecu_custom_cal_absolute_access()`
- `dsecu_custom_cal_paged_access()`
- `dsecu_custom_cal_paged_access_finished()`

For information on the functions, refer to [dSPACE Calibration and Bypassing Service Custom API Functions](#) on page 71.

Note

For a RapidPro system used as a stand-alone prototyping ECU, the custom configuration settings are fixed and already integrated in the dSPACE Calibration and Bypassing Service.

On some ECUs, the DCI-GSI2 cannot access the overlay memory registers via the serial interface. In these cases, the dSPACE Calibration and Bypassing Service is used to perform read and write accesses to the registers. For further information, refer to [DSECU_CAL_MEMORY_COPY_METHOD](#) on page 108.

Integration of custom commands

The dSPACE Calibration and Bypassing Service can be extended by custom-specific commands. A custom command can be added to the service without modifying the service code itself. To implement a customer-specific function, the `dsecu_custom_command` function must be modified.

Note

- For the DCI-GSI2, the custom command feature is available only as part of a dSPACE engineering service, because it requires modifications to the DCI-GSI2's firmware.
- For a RapidPro system used as a stand-alone prototyping ECU, some RapidPro-specific custom commands are already implemented and cannot be modified.

Integration of custom-specific interrupt trigger

By default, an interrupt from the ECU to the tool is triggered by writing to a memory location. In some cases, however, this trigger handling is not possible. In these cases you must implement a custom-specific trigger method by modifying

the `dsecu_custom_trigger_hw_int` function. For example, this function can toggle a digital I/O pin to trigger an interrupt.

Related topics

References

[dSPACE Calibration and Bypassing Service API Functions..... 63](#)

Basics on Service Integration with Multicore ECUs

Introduction

Working with ECU microcontrollers that contain more than one processing core might require integrating the dSPACE Calibration and Bypassing Service into multiple cores.

Service integration into multiple cores

Modern ECU microcontrollers often contain more than one processing core. The dSPACE Calibration and Bypassing Service has to be integrated into multiple cores if one of the following conditions is met:

- The service is to be executed synchronously with tasks running on multiple cores.
- Data that needs to be read/written by the service resides in core local memory, i.e., it can only be accessed by a specific core.

Preparing a single application running on multiple cores

For single applications running on multiple cores, the dSPACE Calibration and Bypassing Service has to be implemented only once. When you integrate a single service instance for multiple cores, you must note the following points:

- The tool RAM area that is used for data exchange between the service and the external tool must be placed in a global RAM that can be accessed by each relevant core.
- Data caching must be disabled for the tool RAM area on each core. (This also applies to running the service on a single core.)
- The `DSECU_SCS_LOCATION` parameter of the service must be set to 1 to make sure that the SCS structure is placed within the tool RAM and can therefore be accessed by all cores. Refer to [DSECU_SCS_LOCATION](#) on page 111.
- To avoid costly mutual exclusion constructs, it is recommended to use the byte-based subinterrupt mechanism.

Alternatively, you can also use bit-based subinterrupts, but you must note the following: The macros `DSECU_INT_SAVE_AND_DISABLE` and `DSECU_INT_RESTORE`, which are used to disable and re-enable the interrupts on the local core, must also implement mutual locks between all cores that execute the dSPACE Calibration and Bypassing Service. Refer to [DSECU_INT_SAVE_AND_DISABLE](#) on page 98 and [DSECU_INT_RESTORE](#) on page 99.

- Service IDs are used to uniquely identify a service call. It is recommended that each identifier is used exclusively by a single core. If this is not possible, you must use mutual exclusion constructs around the service calls to ensure that a service ID is processed by only one core at a time.

Preparing multiple applications running on different cores within the same microcontroller

To provide access to multiple applications running on different cores of one microcontroller, you must note the following points in addition to the points mentioned above:

- The `dsecu_service_init` initialization function must be called once at ECU application startup.
- The `dsecu_service_background` function must be called periodically in a low-priority task.
- Each application must use exactly the same tool RAM addresses and sizes.
- The dSPACE Calibration and Bypassing Service has to be described by only a single A2L file, because it serves as a single interface for accessing all applications.

Related topics

References

[dSPACE Calibration and Bypassing Service API Functions..... 63](#)

Data Structures Used by the dSPACE Calibration and Bypassing Service

Introduction Most of the data structures used by the dSPACE Calibration and Bypassing Service are located in the tool RAM.

Where to go from here

Information in this section

[Basics on the Data Structures](#)..... 34

Most of the data structures used by the dSPACE Calibration and Bypassing Service are located in the tool RAM. These data structures are not shared between the two tools using the dSPACE Calibration and Bypassing Service.

[Service Configuration Section \(SCS\)](#)..... 37

The SCS structure is used to enable or disable data transmission for data acquisition (DAQ) and bypassing, and references further data structures for configuring the dSPACE Calibration and Bypassing Service.

[Table Pointer Section \(TPS\)](#)..... 37

The TPS stores information on interrupt generation and the services which are enabled for the application running on the target ECU.

[Address Tables \(AT\)](#)..... 38

The Address Table consists of an Address Table header and a number of value pairs. Each pair consists of an address entry and an entry for the number of accesses to the address.

[Extended Table Pointer Section \(ETPS\)](#)..... 39

The ETPS stores information for dSPACE Calibration and Bypassing Service calls with IDs > 255.

[Service Pointer Table \(SPT\)](#)..... 39

An SPT stores information associated with one specific dSPACE Calibration and Bypassing Service call with ID > 255.

Basics on the Data Structures

Introduction

Most of the data structures used by the dSPACE Calibration and Bypassing Service are located in the tool RAM. These data structures are not shared between the two tools using the dSPACE Calibration and Bypassing Service. The number of relevant structure types can vary, depending on the maximum number of service IDs used.

Data structures required if `DSECU_MAX_SERVICE_` `COUNT` \leq 255

The following information applies to working with up to 255 dSPACE Calibration and Bypassing Service calls, which means that only service IDs \leq 255 are used.

When started, each tool creates specific data structures which are used by the application running on the target ECU. Three structure types are created in each tool RAM. For details, see:

- [Service Configuration Section \(SCS\)](#) on page 37
- [Table Pointer Section \(TPS\)](#) on page 37
- [Address Tables \(AT\)](#) on page 38

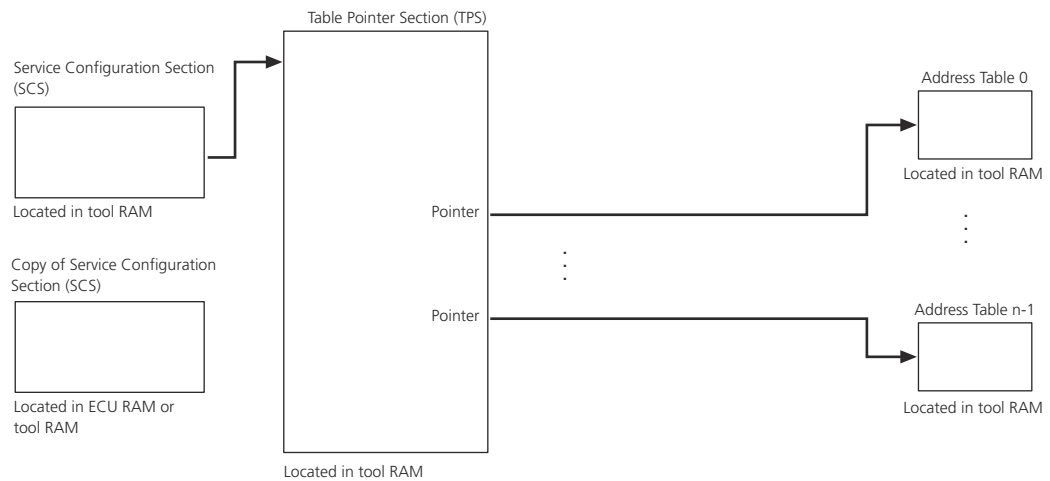
The Service Config Section (SCS) and the Table Pointer Section (TPS) are created only once in each tool RAM.

The AT typically has several entities depending on the number of services used and the amount of data that is transferred between the ECU and each tool.

The SCS contains service enable flags, one for each possible dSPACE Calibration and Bypassing Service call. To avoid flag inconsistency, which can occur if the tool writes to the table while the service accesses it, the dSPACE Calibration and Bypassing Service features a local copy of the SCS in the ECU RAM or in the tool RAM. The `DSECU_SCS_LOCATION` value defines the location of the SCS copies for all tools connected to the ECU. If you want to place the SCS copies in the ECU RAM, the value must be 0. If you want to place the SCS copies in the tool RAM, the value must be 1. Usually, the ECU RAM is faster than the tool RAM, if the tool RAM is located in an external memory (for example, in the dual-port memory of DPMEM-based PODs). If enough memory is available in the ECU RAM, the value can be set to 0 for faster ECU code.

When the ECU application is running, the background task initializes the SCS copy if the corresponding tool is active.

The following illustration shows the data structures in the tool RAM. You can see how the structure types are linked.



Data structures required if DSECU_MAX_SERVICE_ COUNT > 255

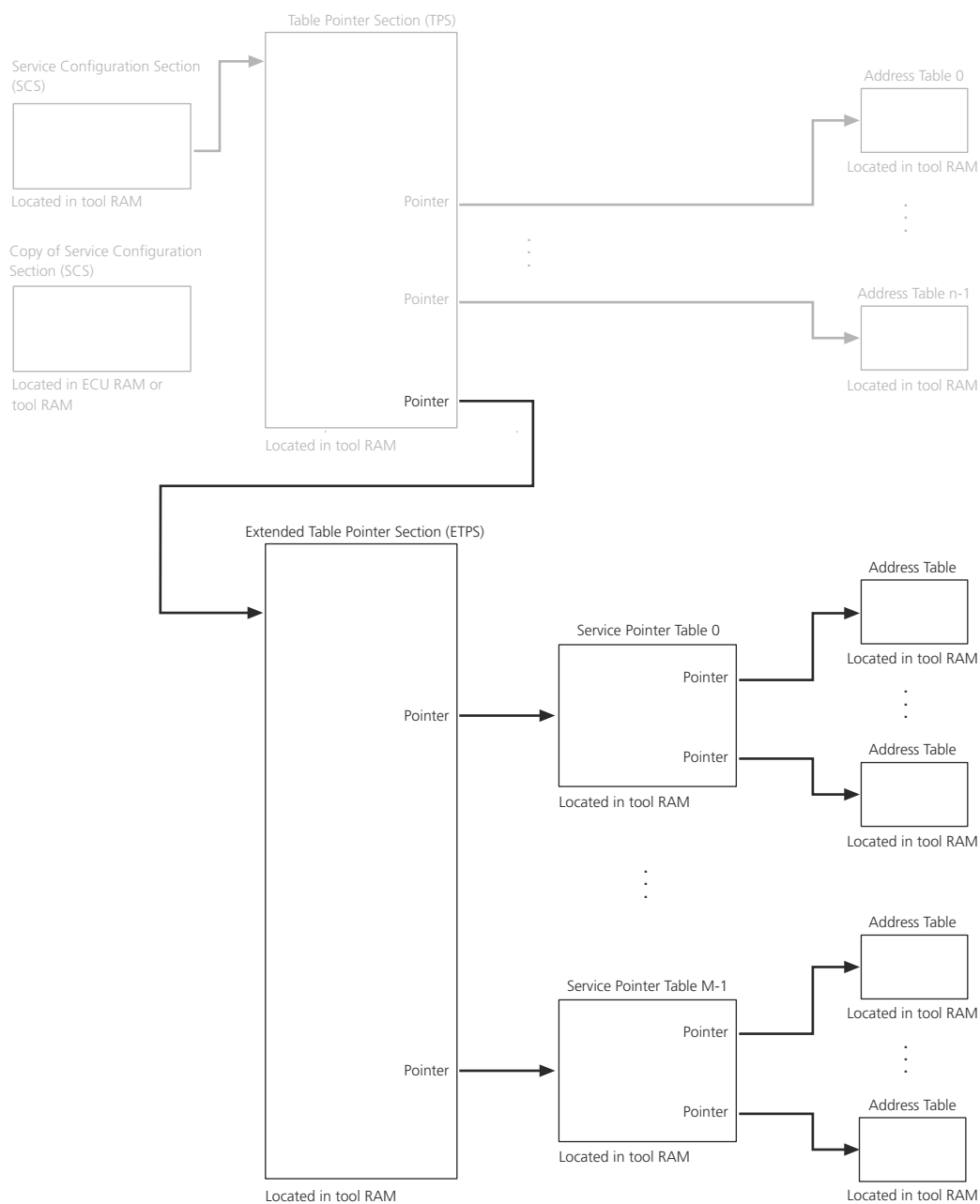
The following information applies to working with more than 255 dSPACE Calibration and Bypassing Service calls, which means that the maximum number of service IDs exceeds 255.

For service calls with service ID ≤ 255 , the data structures are exactly the same as described above (see [Data structures required if DSECU_MAX_SERVICE_COUNT \$\leq 255\$](#) on page 34). For service calls with ID > 255 , two more structure types are required. They are created additionally in each tool RAM when the tool is started. For details, refer to:

- [Extended Table Pointer Section \(ETPS\)](#) on page 39
- [Service Pointer Table \(SPT\)](#) on page 39

The Extended Table Pointer Section (ETPS) is created once in each tool RAM. It contains service enable flags, one for each possible dSPACE Calibration and Bypassing Service call with ID > 255 .

The following illustration shows the data structures that are required if **DSECU_MAX_SERVICE_COUNT** > 255 . Structure types that are used for service calls with ID ≤ 255 are grayed out.



Related topics

References

DSECU_MAX_SERVICE_COUNT..... 127

Service Configuration Section (SCS)

Introduction The SCS structure is used to enable or disable data transmission for data acquisition (DAQ) and bypassing, and references further data structures for configuring the dSPACE Calibration and Bypassing Service.

Service enable flags The SCS features 255 service enable flags for enabling the specific dSPACE Calibration and Bypassing Service calls with ID ≤ 255 . Flag[0], which is the first enable bit (LSB), is used as the global enable flag for all service calls.

- If flag[0] is 0, all service calls are disabled.
- If flag[0] is 1, all service calls with their enable flag set to 1 are enabled.

Note

0 must not be used as a parameter for the `dsecu_service` function.

Related topics	Basics
	Basics on the Data Structures..... 34

Table Pointer Section (TPS)

Introduction The TPS stores information on interrupt generation and the dSPACE Calibration and Bypassing Service calls with service ID ≤ 255 which are enabled for the application running on the target ECU.

For dSPACE Calibration and Bypassing Service calls with service ID > 255 , this information is contained in the Extended Table Pointer Section.

Pointer to Address Table The TPS contains pointer-to-Address-Table entries (32-bit) which point to the corresponding Address Tables.

Each Address Table is assigned to one ECU service call. More than one Address Table can be assigned to one ECU service call at the same time. Since you can configure each Address Table for one data direction, data can be transmitted from and to the ECU within one ECU service call. This is required for some bypassing scenarios, for example.

To use more than one Address Table with one ECU service call, you must enable a functionality definition before compiling the ECU application (see [MORE_ATs_FOR_ONE_SRV](#) on page 112).

**Pointer to Extended Table
Pointer Section**

If DSECU_MAX_SERVICE_COUNT is greater than 255, the TPS additionally contains a pointer to the Extended Table Pointer Section, which holds information on the dSPACE Calibration and Bypassing Service calls with IDs > 255.

Related topics**Basics**

[Basics on the Data Structures.....](#) 34

References

[DSECU_MAX_SERVICE_COUNT.....](#) 127

Address Tables (AT)

Introduction

The AT consists of an Address Table header and a number of value pairs. Each pair consists of an address entry and an entry for the number of accesses to the address.

Value pairs for block transfer

If the block transfer mode is active (see [Block Copy Mechanism](#) on page 58), the number-of-accesses entries are used to determine the size of the block. The entries specify the number of accesses with the selected data type, so the size of the data block (in bytes) is calculated by multiplying the number of accesses by the size of the selected data type. Access starts with the address contained in the corresponding start address entry.

Example

For example, if the address in the start address entry for 32-bit accesses is 0x1000 and the number of 32-bit accesses is 3, the addresses used for data transfers are 0x1000, 0x1004, 0x1008.

Value pairs for bit transfer

For bit transfers, the number-of-accesses entry does not specify the number of accesses. A bit mask for specifying the bits needed to be modified is given instead.

- First all bits with a mask bit of 0 are cleared. The other bits remain unchanged.
- Then all bits are set additionally whose corresponding bit in the value to be written is a 1 bit.

Note

In the RTI Bypass Blockset, the bit mask procedure differs from that described above.

The tool must guarantee that only bits that are to be modified are set in the tool RAM. The mask operation is done only if the ECU reads data from the tool RAM. Bit operations are not supported for the ECU-to-tool direction. 8-bit operations must be used instead. The ECU writes the complete 8-bit value to the tool RAM. The tool masks and shifts the received 8-bit value in order to get the values of the relevant bits.

Related topics

Basics

[Basics on the Data Structures..... 34](#)

Extended Table Pointer Section (ETPS)

Introduction

The ETPS stores information for dSPACE Calibration and Bypassing Service calls with IDs > 255. The ETPS is only present if DSECU_MAX_SERVICE_COUNT is greater than 255.

Pointer to Service Pointer Table

The ETPS contains enable flags for the possible dSPACE Calibration and Bypassing Service calls with service ID > 255. It also contains pointer-to-Service-Pointer-Table entries which point to the corresponding Service Pointer Tables.

Each Service Pointer Table is assigned to exactly one active service call with ID > 255.

Related topics

Basics

[Basics on the Data Structures..... 34](#)

References

[DSECU_MAX_SERVICE_COUNT..... 127](#)

Service Pointer Table (SPT)

Introduction

An SPT stores information associated with one specific dSPACE Calibration and Bypassing Service call. SPTs are used only for service calls with ID > 255.

For dSPACE Calibration and Bypassing Service calls with ID ≤ 255 , this information is held in the Table Pointer Section.

Pointer to Address Table

An SPT contains a list of pointers to the Address Tables that have to be processed in the service call.

Each Address Table is assigned to one service call. More than one Address Table can be assigned to one ECU service call at the same time.

Related topics

Basics

[Basics on the Data Structures..... 34](#)

dSPACE Calibration and Bypassing Service Mechanisms

Where to go from here

Information in this section

Extended Bypassing Mechanisms.....	41
The dSPACE Calibration and Bypassing Service provides some bypassing mechanisms to ensure operational reliability. For information on these mechanisms and their dependencies.	
Alive and Version Information Mechanism.....	47
The alive and version information mechanism is used to detect if a calibration tool or bypassing system is connected to the ECU and is running.	
Subinterrupt Handling Mechanism.....	55
The subinterrupt handling mechanism generates and handles multiple subinterrupts using a single hardware interrupt line.	
Block Copy Mechanism.....	58
The block copy mechanism allows you to transfer data blocks.	
ECU Flash Programming Mechanism.....	59
The dSPACE Calibration and Bypassing Service can be used for ECU flash programming.	

Extended Bypassing Mechanisms

Extended bypassing mechanisms

- The dSPACE Calibration and Bypassing Service provides extended bypassing mechanisms:
- Double buffer mechanism
 - Wait mechanism

- Buffer synchronization mechanism
- Failure checking mechanism
- Fail-safe mechanism

These mechanisms are interdependent. The dSPACE Calibration and Bypassing Service provides specific commands for configuring them.

Double buffer mechanism

The double buffer mechanism is used to ensure that only consistent data blocks are transmitted. If the mechanism is enabled, two buffers are used for data exchange between the ECU and the RCP (rapid control prototyping) system. For example, when reading data from the RCP system, the ECU continues reading old data (= data already read before) as long as no new data is available from the RCP system. This ensures data consistency. If you disable the double buffer mechanism, only one buffer is used for data exchange.

The dSPACE Calibration and Bypassing Service supports the double buffer mechanism. You can use the configuration file to specify whether the mechanism is to be supported. Then you must enable or disable the double buffer mechanism explicitly in every Read and Write block of the RTI Bypass Blockset.

If the double buffer mechanism is disabled, the wait, failure checking, and fail-safe mechanisms are not available.

Wait mechanism

You can use the wait mechanism to enable the ECU to wait for a valid response from the RCP (Rapid Control Prototyping) system. If the mechanism is enabled, the ECU waits until new data is available from the RCP system. If the ECU receives no new data until a predefined timeout, the ECU continues working in a defined way. The wait mechanism ensures that the ECU uses only data calculated by the RCP system.

The dSPACE Calibration and Bypassing Service supports the wait mechanism. You can use the configuration file to specify whether the mechanism is to be supported. Then you must enable or disable the wait mechanism explicitly for every event channel (service instance) in the Write blocks of the RTI Bypass Blockset.

Note

- You can enable the consistency wait mechanism only if you also enable the double buffer mechanism.
- Using the wait mechanism may increase the ECU's task execution time.
- You must configure a timeout scaling factor in the service configuration or alternatively realize a custom time stamp functionality within the service customization layer to ensure that the effective waiting time and the wait time configuration match. For further information on the functionality and configuration of the different timeout detection methods, refer to [Configuration of DAQ and Bypassing Features](#) on page 110.

Buffer synchronization mechanism

The buffer synchronization mechanism is used to enable the ECU to recognize whether data that is written back from the RCP system to the ECU belongs to the current sampling step, or whether it is delayed data from the previous sampling step. Data that is written back from the RCP system to the ECU after the ECU waiting time (specified in the wait mechanism) has expired is marked as old data. This prevents old results from being erroneously interpreted as current results.

Delayed data is identified by comparing identifier values of read and write service calls. When sending data to the RCP system, the ECU also transmits a unique identifier to the RCP system. (Each time a write service is called, the identifier value is incremented.) When the RCP system writes the calculated data back to the ECU, it also returns the identifier it received from the ECU. When new data is available from the RCP system, the ECU compares its current local identifier with the identifier received from the RCP system. Matching identifiers indicate results of the current sampling step. If the identifiers differ, the received data is immediately marked as old data (by setting old buffer equal to current buffer), and the ECU continues waiting for new data.

The dSPACE Calibration and Bypassing Service supports the buffer synchronization mechanism. You can use the dSPACE Calibration and Bypassing Service API function to activate the mechanism.

Note

- As of RTI Bypass Blockset 2.1, the buffer synchronization settings can be set conveniently in the blockset. The `dsecu_service_buffer_sync` function is no longer required.
- The buffer synchronization mechanism is active only if the wait mechanism is enabled.
- The buffer synchronization mechanism applies to data transmission only from the RCP system to the ECU.

Failure checking and fail-safe mechanisms

The dSPACE Calibration and Bypassing Service lets you enable the failure checking and the fail-safe mechanisms.

Failure checking mechanism The failure checking mechanism is used by the read service call for counting the number of failed data exchanges. A data exchange failure occurs when the ECU has not received new data from the rapid control prototyping (RCP) system. The failure checking mechanism must be enabled or disabled globally via the configuration file.

Fail-safe mechanism The fail-safe mechanism provides valid data (fail-safe data) if the RCP system has failed to provide new data to the ECU more often than indicated in the failure limit value.

Whenever the ECU receives no new data from the RCP system, the failure number is incremented. When the failure number reaches the predefined failure

limit, the following happens (depending on the configuration settings made in the RTI Bypass Blockset and on the `DSECU_FAILSAFE_PAGE` define):

- If the fail-safe mechanism is enabled and if a fail-safe page (memory area containing data to be used by the ECU) exists, the ECU uses the data from the fail-safe page for further calculation.
- If the fail-safe mechanism is not enabled, the service returns **no data copied**, and the ECU uses the data calculated by its own control algorithm.

If the RCP system sends new data to the ECU, the failure count is reset to 0. The fail-safe mechanism ensures that the ECU continues working with “safe” data even if, for example, the connection between the ECU and the RCP system is interrupted.

The dSPACE Calibration and Bypassing Service supports the fail-safe mechanism. You can use the configuration file to specify whether the mechanism is to be supported. Then you must enable or disable the fail-safe mechanism explicitly for every event channel (service instance) in the Write blocks of the RTI Bypass Blockset.

Note

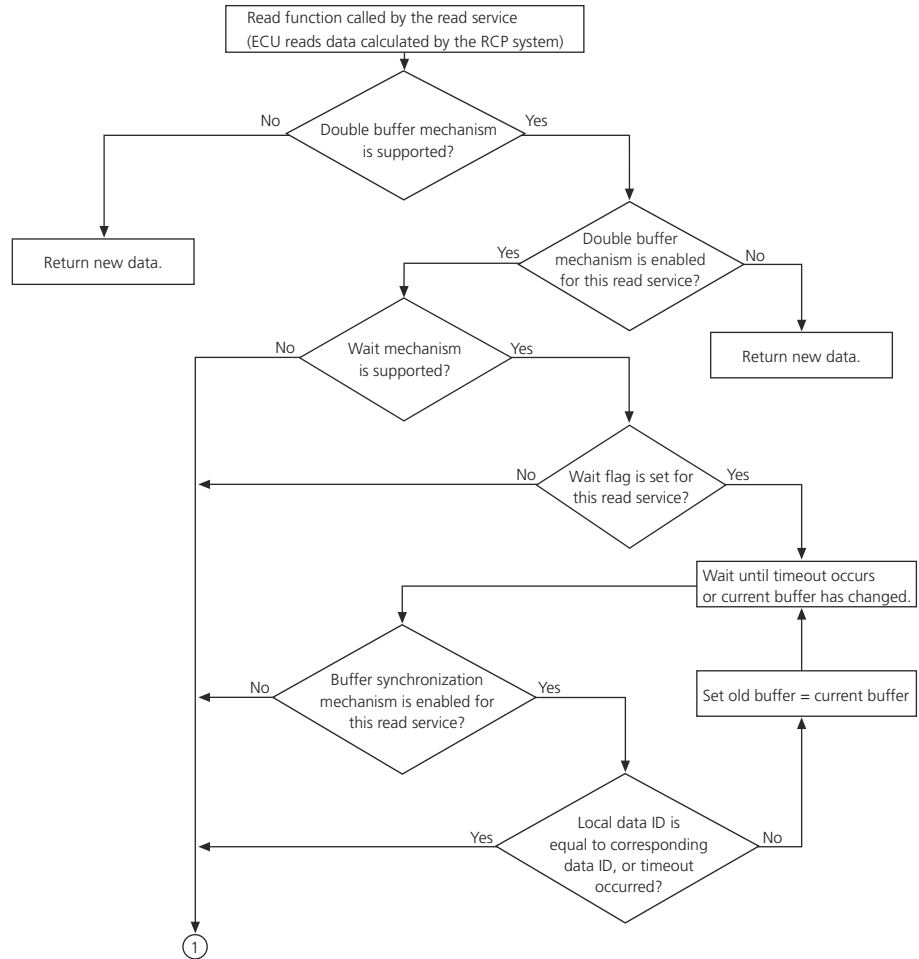
The fail-safe mechanism is currently not supported by the RTI Bypass Blockset.

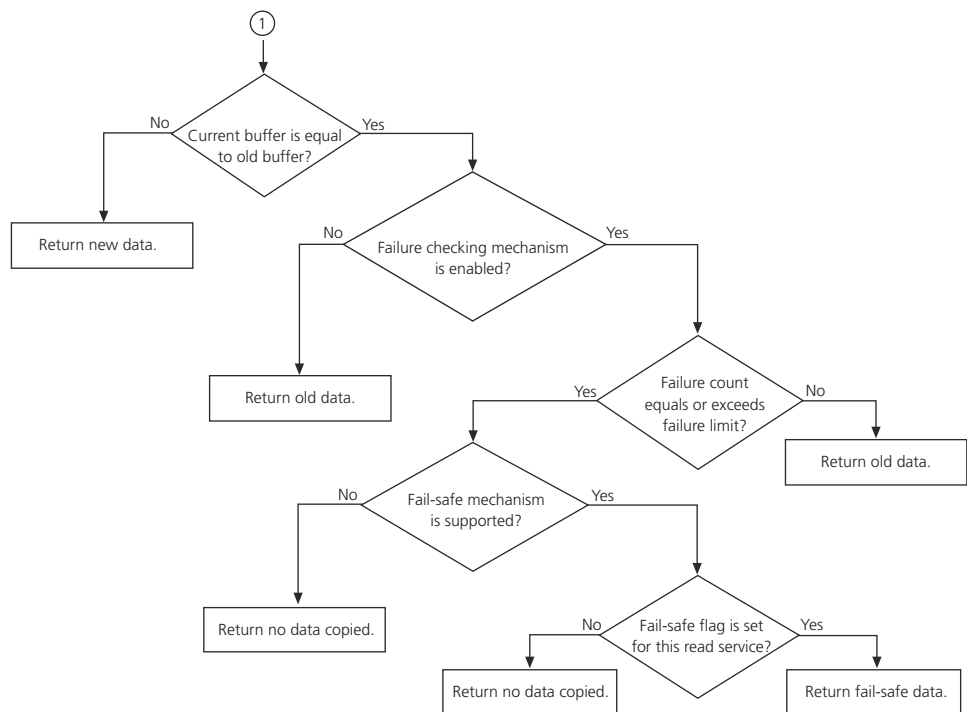
Note

- You can enable the failure checking and fail-safe mechanisms only if you also enable the double buffer mechanism.
- To make the fail-safe mechanism available, the failure checking mechanism must be enabled.

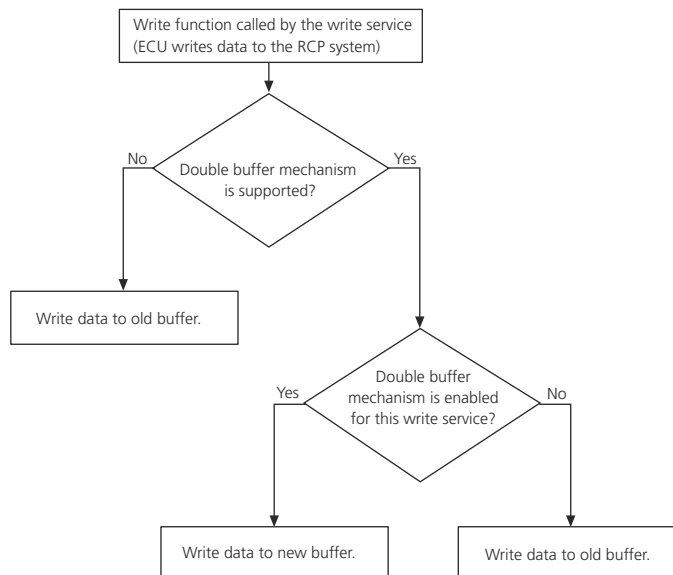
Flowcharts of extended bypassing mechanisms

Read service call The following illustrations show the extended bypassing mechanisms integrated for the read service call:





Write service call The following illustration shows the extended bypassing mechanisms integrated for the write service call:



Note

Some of the decisions shown in the illustrations above must be made in the configuration file (for example, whether the double buffer mechanism is supported). Other settings must be made on the dialog pages of the RTI Bypass Blockset.

Related topics

References

[Configuration of DAQ and Bypassing Features.....](#) 110

Alive and Version Information Mechanism

Introduction

The dSPACE Calibration and Bypassing Service uses the alive and version information mechanism to check if a calibration tool (ControlDesk) and/or bypassing system (for example, DS1006 or MicroAutoBox II) is connected to the ECU and is running.

Note

The alive and version information mechanism is always enabled and cannot be configured or adapted.

Memory location

Because measurement and bypassing in parallel is supported, the alive and version information mechanism addresses two different memory ranges. The locations used for communication between the tool and the ECU are placed in each tool RAM, starting with the address defined by `DSECU_TOOL_30_PLUS_X_FREE_DPM_WORDS(tool_nr)` (see `dsecusvc.h`).

The following definitions are used for the alive and version information mechanism (see also the illustration below):

```
#define DSECU_VERSION_L          (0)
#define DSECU_VERSION_H          (1)
#define DSECU_COMPILED_FOR       (2)
#define DSECU_ALIVE_DEVICE_WRITE (3)
#define DSECU_ALIVE_ECU_WRITE    (4)
#define DSECU_ALIVE_DEVICE_READ  DSECU_ALIVE_ECU_WRITE
#define DSECU_ALIVE_ECU_READ     DSECU_ALIVE_DEVICE_WRITE
#define DSECU_DEVICE_STILL_ALIVE (5)
#define DSECU_ECU_STILL_ALIVE    (6)
#define DSECU_RESERVED_WORD      (7)
#define DSECU_MAGIC_WORD          (8)
#define DSECU_DEVICE_VERSION_L   (0xA)
#define DSECU_DEVICE_VERSION_H   (0xB)
#define DSECU_MAILBOX_ADDR        (0xC)
#define DSECU_DEVICE_RESET_COUNTER (0xE)
#define DSECU_ECU_RESET_COUNTER  (0xF)
#define DSECU_LOW_LEVEL_CHECK     (0x10)
#define DSECU_SERVICE_COUNT       (0x12)
#define DSECU_SUBINT_INFO         (0x13)
#define DSECU_FEATURE_FLAGS0      (0x14)
#define DSECU_FEATURE_FLAGS1      (0x15)
#define DSECU_MASTER_FEATURE_LEVEL (0x16)
```

- **DSECU_VERSION_L**, **DSECU_VERSION_H** and **DSECU_COMPILED_FOR** are used for compatibility checks. They contain the version of the dSPACE Calibration and Bypassing Service code (low and high word), and a flag that indicates the RCP hardware the service code was compiled for.
- **DSECU_ALIVE_DEVICE_WRITE** and **DSECU_ALIVE_ECU_WRITE** are used to determine whether the other side is alive. The first word is written by the tool and read by the ECU, the second word is written by the ECU and read by the tool.
- **DSECU_DEVICE_STILL_ALIVE** and **DSECU_ECU_STILL_ALIVE** are used to indicate that the tool or ECU side is switched off. During normal operation a magic word is written to these memory locations. If one side shuts down, it writes any other value to the appropriate location to inform the other side.
- **DSECU_DEVICE_RESET_COUNTER** and **DSECU_ECU_RESET_COUNTER** words are used to detect a reset. If an ECU reset occurs or if the application running on the RCP system is restarted, the corresponding counter is incremented.
- The 32-bit word **DSECU_LOW_LEVEL_CHECK** is used by the tool to check whether memory accesses to the tool RAM area are working correctly.
- **DSECU_SERVICE_COUNT** indicates the maximum number of service IDs supported by the service. The value is determined by setting the **DSECU_MAX_SERVICE_COUNT** configuration option.
- **DSECU_SUBINT_INFO**, **DSECU_FEATURE_FLAGS0** and **DSECU_FEATURE_FLAGS1** contain information on the configuration of the dSPACE Calibration and Bypassing Service. This information can be used by the tool to detect misconfigurations.
- **DSECU_MASTER_FEATURE_LEVEL** is used to tell the service which features are supported by the tool.

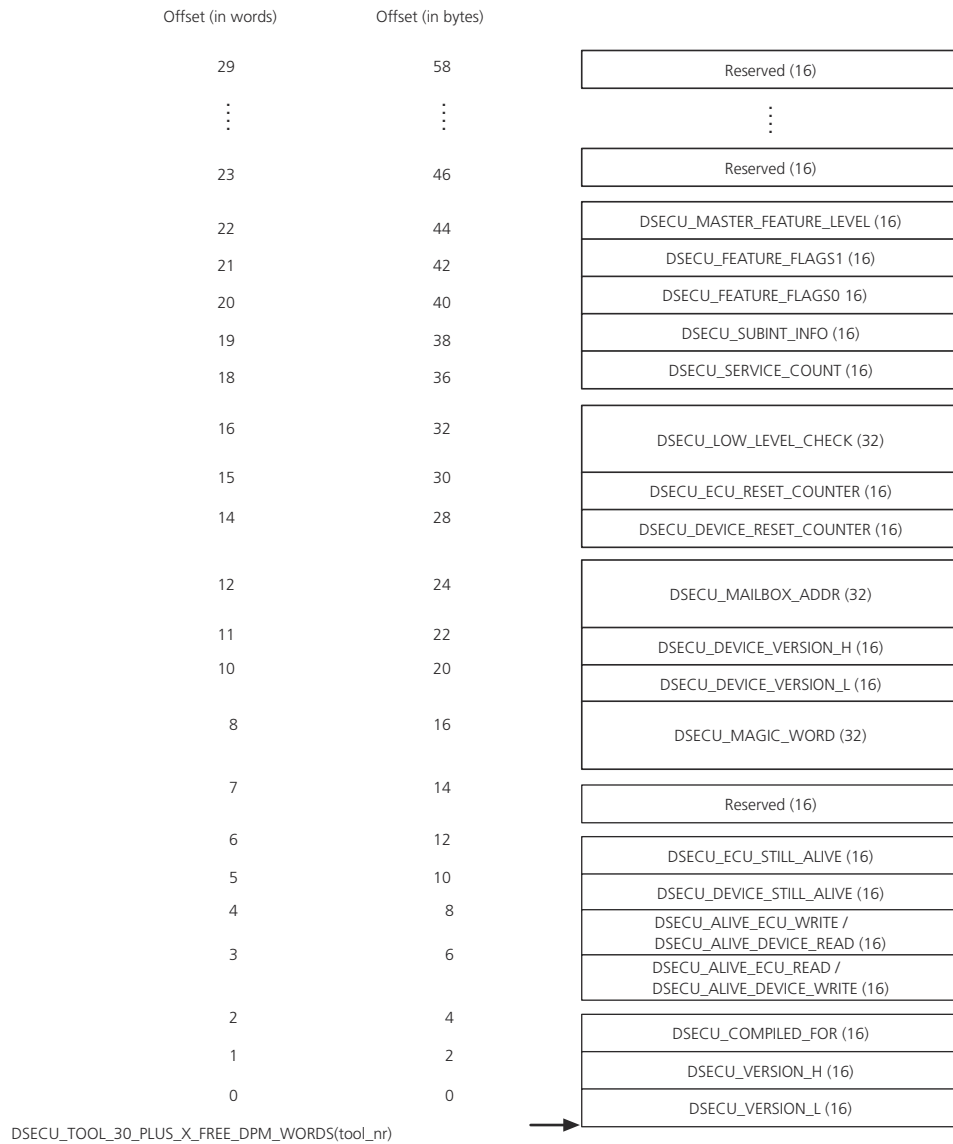
The following words are used only if any feature besides data acquisition and bypassing is enabled in the service.

- **DSECU_MAGIC_WORD** is a 32-bit value that is set by the tool (only dSPACE ECU interface) to indicate that the (command) mailbox is present. The **DSECU_RESERVED_WORD** entry is inserted to align the magic word to an address divisible by 4.
- **DSECU_DEVICE_VERSION_L** and **DSECU_DEVICE_VERSION_H** are used for compatibility checks on the ECU. They are written by the tool (only dSPACE ECU interfaces) and contain the version of the tool software with respect to the dSPACE Calibration and Bypassing Service (low and high word).
- The **DSECU_MAILBOX_ADDR** word specifies the address of the mailbox. All commands and responses between the tool (only dSPACE ECU interface) and the ECU are sent via the mailbox.

Note

The start address of the first **DSECU_MAILBOX_ADDR** entry is always 32-bit aligned.

The following illustration shows the alive and version information words in the tool RAM:



ECU initialization

When the dSPACE Calibration and Bypassing Service starts, it first clears the still alive flag (**DSECU_ECU_STILL_ALIVE**) to signal to the tool that the ECU is not alive (see illustration below).

The tool alive counter is read (**DSECU_ALIVE_ECU_READ**) and stored locally. Its value is used to check whether the tool alive counter has changed. For example, it is incremented during initialization of the RCP application.

The **DSECU_SINT_INIT** subsystem (which is actually a macro in the C code) is passed through. If bit-based subinterrupt handling is enabled, the valid bit in the

subinterrupt request word(s) is cleared to prevent subinterrupts being requested by mistake. All subinterrupt request words are initialized to 0, before the ECU alive counter is incremented to signal to the tool that the ECU is alive.

Back in the initialization function, the ECU reset counter (DSECU_ECU_RESET_COUNTER) is incremented.

Finally, the ECU still alive flag is set again.

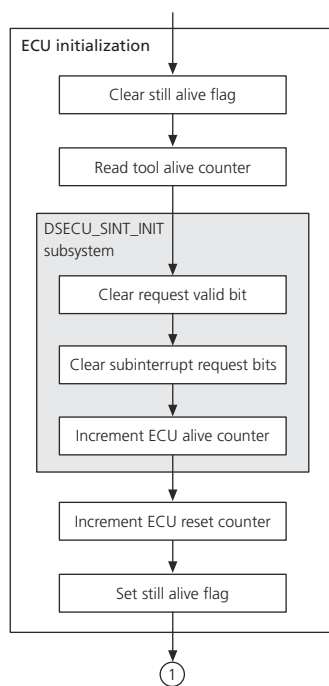
ECU background

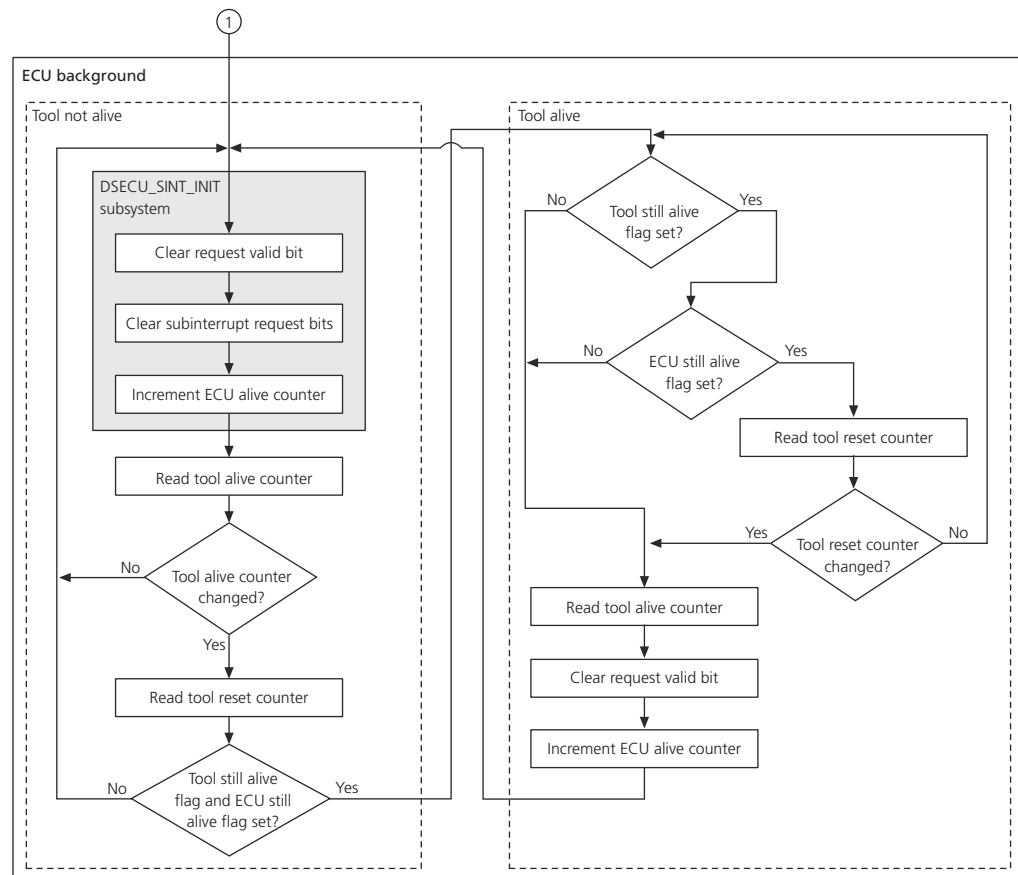
After ECU initialization, the ECU repeatedly calls the background function of the dSPACE Calibration and Bypassing Service. Other actions depend on the current state of the tool.

- If the tool was not yet detected as alive, the DSECU_SINT_INIT subsystem is called again. This checks whether the tool alive counter has been incremented. If not, the background function is left, and the tool is not detected as alive. If the tool alive counter has been changed (DSECU_ALIVE_DEVICE_WRITE), the tool reset counter (DSECU_DEVICE_RESET_COUNTER) is read and stored locally. This value is used to check whether the tool reset counter has changed. For example, it is incremented when the tool is restarted. Finally, both the ECU still alive flag and the tool still alive flag are checked. The tool is recognized as alive only if both still alive flags are set correctly.

The tool's background function checks whether the ECU alive counter has changed. This is the case if the background function of the ECU has been called at least once. The tool alive counter is now incremented, and the ECU is identified as alive.

- If the tool is recognized as alive, the settings of the tool still alive flag and the ECU still alive flag are checked, that is, the DSECU_DEVICE_STILL_ALIVE and DSECU_ECU_STILL_ALIVE words are checked to determine if the tool and ECU are still connected and running. If any check is negative, the tool is assumed to be not alive, and the alive mechanism is restarted. If the checks are positive, the tool reset counter (DSECU_DEVICE_RESET_COUNTER) is read. If the value has changed, the tool has been restarted, and the alive mechanism must be restarted. The tool alive counter is stored locally as a compare value for the alive check. The subinterrupt request bits are invalidated by clearing the valid bit (for bit-based subinterrupt handling only), and the ECU alive counter is incremented. In the next call to the background function, the alive mechanism acts as if the tool is not alive.





Tool initialization

The tool part of the alive mechanism is similar to the ECU part, except that the use of counters is reversed.

During tool initialization, the ECU alive counter is read for later comparison with the recent value in the background function. If the ECU increments the counter in the meantime, the tool proceeds to the alive state in the first call to the background function.

If bit-based subinterrupt handling is enabled, the valid bit of the ready words is cleared.

Finally, the tool alive counter and the tool reset counter are incremented.

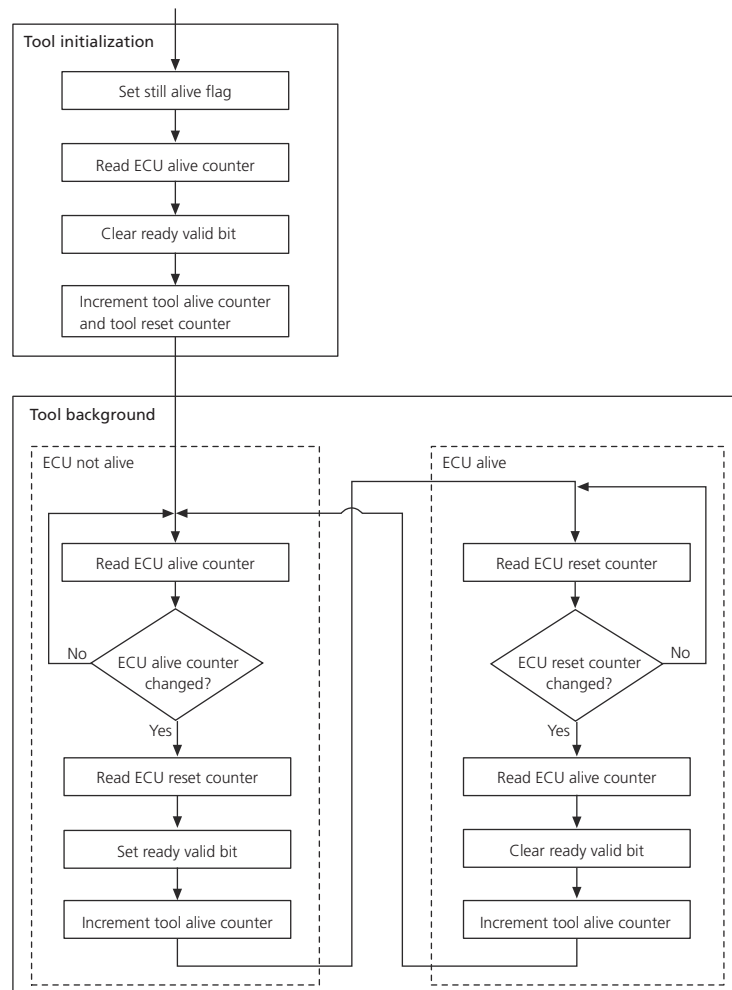
Tool background

The behavior of the background function depends on the current alive state of the ECU.

- If the ECU is not alive, the ECU alive counter is checked for a change. If it is unchanged, the background function is left and the ECU is not detected as alive. If the ECU alive counter has been changed, the ECU is detected as alive. The ECU reset counter is read and the value is stored for later comparison.

Then the valid bit in the ready words is set for triggering the subinterrupts. Finally, the tool alive counter is incremented, and the ECU state changes to alive.

- If the ECU was already detected as alive, in every call to the background function the ECU reset counter is read and compared to the value that was stored when the ECU changed to the alive state. If the counter is unchanged, the dSPACE Calibration and Bypassing Service operates as normal. If the ECU reset counter has changed (that is, the ECU has been reset), the alive mechanism must be restarted. For this reason the ECU alive counter is read and stored locally as it is in the initialization function. To avoid further subinterrupts being triggered, the valid bit in the ready words is cleared. Finally, the tool alive counter is incremented to signal that the tool is alive.



Related topics**Basics**

[Bypassing Features \(dSPACE Calibration and Bypassing Service Feature Reference !\[\]\(feabb98897b440bc8695a03336a6e2df_img.jpg\)\)](#)
[Measurement Features \(dSPACE Calibration and Bypassing Service Feature Reference !\[\]\(c7f935293d8062fa748ed86b74d28761_img.jpg\)\)](#)

Subinterrupt Handling Mechanism

Introduction

The subinterrupt handling mechanism generates and handles multiple subinterrupts using a single hardware interrupt line. The locations involved in subinterrupt generation are located in each tool RAM connected to your target ECU.

Subinterrupt handling methods

The dSPACE Calibration and Bypassing Service supports three different methods of subinterrupt handling:

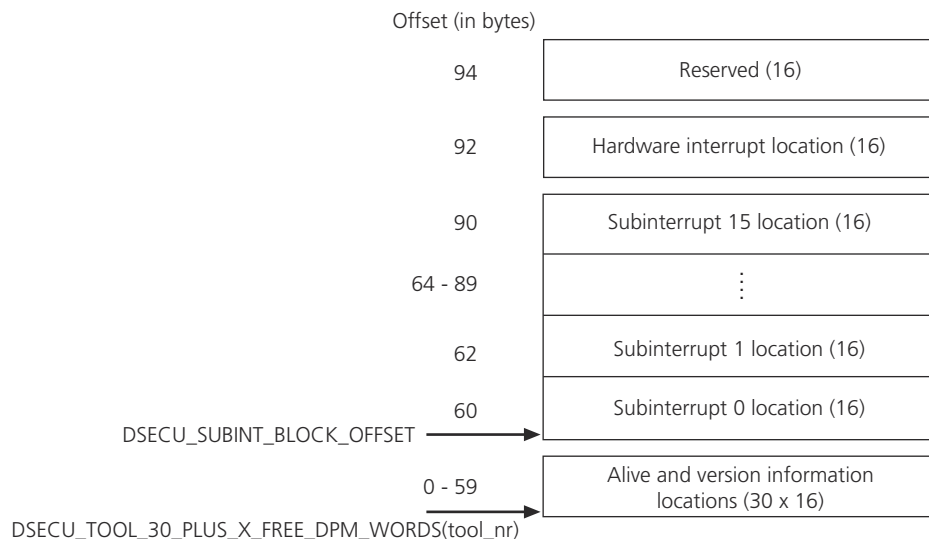
- **Byte-based subinterrupt handling**
 With byte-based subinterrupt handling, one byte is used per subinterrupt. For information on this method, refer to [Byte-Based and Word-Based Subinterrupt Handling](#) on page 55.
- **Word-based subinterrupt handling**
 With word-based subinterrupt handling, two bytes are used per subinterrupt. For information on this method, refer to [Byte-Based and Word-Based Subinterrupt Handling](#) on page 55.
- **Bit-based subinterrupt handling**
 With bit-based subinterrupt handling, one bit is used per subinterrupt. For information on this method, refer to [Bit-Based Subinterrupt Handling](#) on page 56.

Byte-Based and Word-Based Subinterrupt Handling

The mechanisms for byte-based and word-based subinterrupt generation are very similar. The difference is that in the one case only one byte is written to/read from the tool RAM instead of two bytes.

Byte- or word-based subinterrupt handling locations The first subinterrupt handling location starts at offset `DSECU_SUBINT_BLOCK_OFFSET` from the address `DSECU_TOOL_30_PLUS_X_FREE_DPM_WORDS(tool_nr)`, where the alive and version information locations begin for the tool indicated by `tool_nr`. For example, if 16 subinterrupts are used, each of the first 16 locations (words or bytes) corresponds to one subinterrupt.

The following illustration shows the locations involved in word-based subinterrupt generation for 16 subinterrupts:



The locations for byte-based subinterrupt generation are only one byte long. Hence, the memory space needed for byte-based subinterrupt handling is smaller. The width of the hardware interrupt word is 16 bits in any case.

Subinterrupt handling method On start-up, the service code initializes all subinterrupt locations by `DSECUCUBINT_STATE_SINT_READY_DATA_INVALID` (subinterrupt state: ready, data: invalid). To trigger a subinterrupt to the tool, the service code writes the value `DSECUCUBINT_STATE_SINT_PENDING` (subinterrupt: pending) to the specific subinterrupt location, and triggers a hardware interrupt line by writing `DSECUCUBINT_STATE_SINT_PENDING` to the hardware interrupt location for the tool given by `tool_nr`. When the tool has finished processing the subinterrupt, it clears the subinterrupt location by writing `DSECUCUBINT_STATE_SINT_READY` to it, and releases the hardware interrupt in the same way.

Bit-Based Subinterrupt Handling

Like byte-based and word-based subinterrupt handling, the implementation of the bit-based subinterrupt handling also requires three states.

Bit categories However, for bit-based subinterrupt handling, each state is split into three bits:

- A "trigger bit"
- A "ready bit"
- An "acknowledge bit"

These bits are located in three different words:

- Trigger bits in the "subinterrupt pending word(s)".
- Ready bits in the "subinterrupt ready word(s)".
- Acknowledge bits in the "subinterrupt acknowledge word(s)".

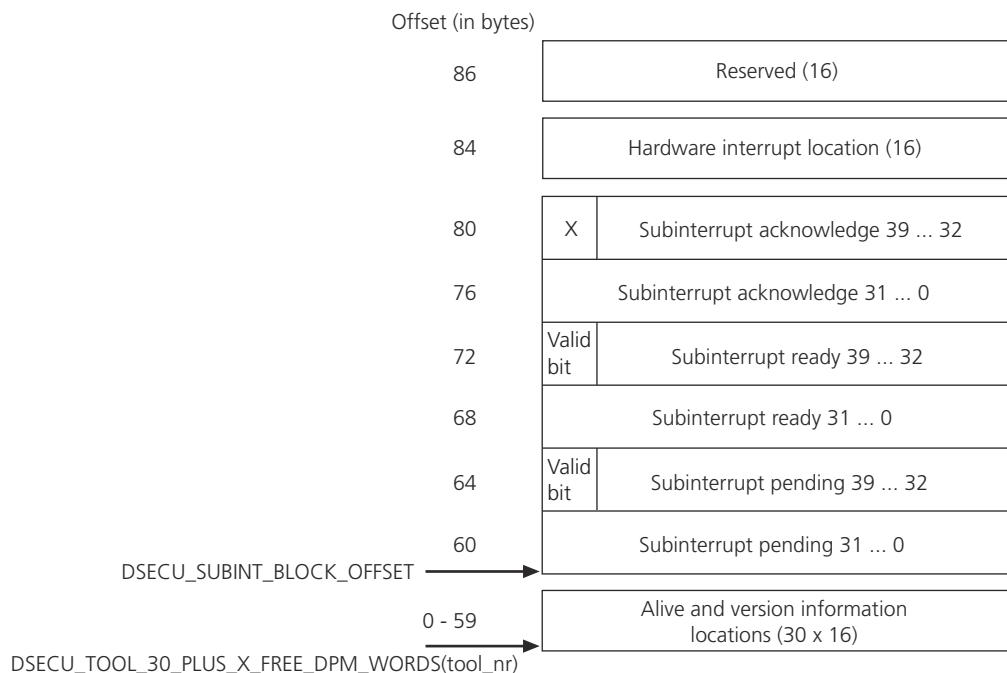
Subinterrupt handling locations The first subinterrupt pending word starts at offset `DSECUCUBINT_BLOCK_OFFSET` from the address

`DSECUC_TOOL_30_PLUS_X_FREE_DPM_WORDS(tool_nr)`, where the alive and version information locations begin for the tool indicated by `tool_nr`. It is followed by further subinterrupt pending words (if existent), the subinterrupt ready word(s) containing the ready bits, and the subinterrupt acknowledge word(s) containing the acknowledge bits (see illustration below).

For each bit category, the bits are assigned to the subinterrupts. Starting with the least significant bit (LSB) of the first word, the bits are assigned to subinterrupt 0, subinterrupt 1,

For a number of subinterrupts of up to 31, one subinterrupt pending word, one subinterrupt ready word, and one subinterrupt acknowledge word are required. If the number of subinterrupts is between 32 and 63, two words are needed for each bit category. If more subinterrupts are needed, a proportionate number of additional words are reserved for each bit category.

The following illustration shows an example of bit-based subinterrupt handling using 40 subinterrupts.



Valid bits Additionally, a valid bit is included in the MSB of the last subinterrupt pending word and the last subinterrupt ready word. The corresponding bit in the last subinterrupt acknowledge word is left void.

The valid bits determine whether the other bit entries are valid or not:

- Valid bit of subinterrupt pending word is 0
All other bit entries of the pending memory area are invalid.
- Valid bit of subinterrupt pending word is 1
All other bit entries of the pending memory area are valid.
- Valid bit of subinterrupt ready word is 0
All other bit entries of the pending and acknowledge memory areas are invalid.

- Valid bit of subinterrupt ready word is 1

All other bit entries of the pending and acknowledge memory areas are valid.

Bit-based subinterrupt handling method To submit a subinterrupt, the valid bit of the subinterrupt pending word must be set to 1. Then the corresponding trigger bit is toggled, and a hardware interrupt is triggered. The tool receives the hardware interrupt and reads the pending words, starting by checking the valid bit of the subinterrupt ready word. If the valid bit is 0, the ready words are not valid, and no subinterrupt is triggered.

In some cases, a tool uses only the subinterrupt pending words and the subinterrupt ready words to manage subinterrupt handling. The bits of the pending and ready words are compared. A varying bit indicates that the corresponding subinterrupt was triggered. Data acquisition for the corresponding channel is performed, and finally, the ready bit is toggled.

If the tool uses the three word categories, the subinterrupt pending words must be checked against the subinterrupt ready and subinterrupt acknowledge words. If a trigger bit differs from both the associated ready bit and the associated acknowledge bit, the subinterrupt is triggered and the tool toggles the acknowledge bit. In this state, the ECU is not allowed to trigger this subinterrupt again. The tool processes the subinterrupt and finally toggles the ready bit.

Related topics

Basics

[Alive and Version Information Mechanism..... 47](#)

Block Copy Mechanism

Introduction

The block copy mechanism is used in the service call for transferring data blocks.

Block copy mechanism

If the block copy mechanism is enabled, all data transfers use the address given in the address entry of the Address Table (AT) as the start address. The number of addresses included in the data transfer is written to the number-of-accesses entry of the AT, corresponding to the address entry.

Example For the start address 0x1000 and a block of 3 data items with 32-bit width, the calculated addresses are 0x1000, 0x1004 and 0x1008.

The addresses are calculated from the start address by incrementing by a value depending on the data type used in the transfer:

- 8 for 64-bit data transfers (number of bytes in 64-bit data)
- 4 for 32-bit data transfers
- ...

Example Suppose a data block of 16 bytes is to be transferred. If the support of 64-bit data types is enabled, one entry is generated in the AT containing the start address, and the number-of-accesses entry corresponding to the address entry is set to 2.

If the block copy mechanism is disabled, only the address given in the address entry of the Address Table (AT) is used. The number-of-accesses entry of the AT corresponding to the address entry is ignored. If a data block has to be transferred, several entries are generated in the AT.

Related topics

Basics

[Measurement Features \(dSPACE Calibration and Bypassing Service Feature Reference !\[\]\(a03a7eb2f4046e1d3c76772003e549ea_img.jpg\)](#))

ECU Flash Programming Mechanism

Introduction

The dSPACE Calibration and Bypassing Service supports ECU flash programming for all kinds of ECUs and flash devices.

General ECU flash programming concept

When configuring the dSPACE Calibration and Bypassing Service, you can split the flash memory of an ECU into segments. For example, you can define different segments for the ECU application code, the boot code, the calibration data, and even the microcontroller's shadow configuration. Custom-specific functions must be integrated into the service for erasing and programming the individual segments.

Each segment defines an area of the flash memory which may be accessed by the flash tool. The flash tool can flash each segment independently, for example, if a calibrated data set is to be flashed to the ECU, while the segments containing ECU code must remain unchanged. Flash areas which are not covered by a segment definition will not be modified by the flash tool.

The dSPACE Calibration and Bypassing Service also supports flashing memory ranges which are normally not mapped to the ECU's linear address range (for example, flashing an external I2C bus flash device or a processor's internal shadow configuration). In this case the custom erase and program functions can be adapted to map a specific memory device to an unused address range of the ECU. This address range can then be used by the flash tool.

Sequence numbers for erasing and writing are defined for each individual segment. You can therefore control the exact sequence in which the segments are programmed to the ECU.

The number of segments, their absolute addresses, and the sequence numbers for erasing/writing are part of the dSPACE Calibration and Bypassing Service.

They are compiled in the flash kernel. Thus, the flash tool can read these configuration settings directly from the ECU.

The following illustration shows an example of a flash memory layout:

Flash device	Address range in ECU memory	Sector	Address range used by the flash tool and dSPACE Calibration and Bypassing Service
Internal flash	0x0 - 0x1FFF 0x2000 - 0x7FFFF	Sector 0 Boot code	0x0 - 0x1FFF 0x2000
		Sector 1 ECU application	- 0x7FFFF
External flash	0x400000 - 0x40FFFF	Sector 2 Calibration data	0x400000 - 0x40FFFF
Internal configuration	No absolute address	Sector 3 Shadow configuration	0xFFFF0000 - 0xFFFF00FF

Command sequence for ECU flash programming

For ECU flash programming, the flash tool uses the following flash commands:

- To signal the beginning of a flash sequence, the `dsecu_flash_program_start` command is called.
- A series of `dsecu_flash_erase_range` and `dsecu_flash_program_range` commands is used to erase/program the flash memory.
- `dsecu_flash_program_finish` is called to signal the end of the flash sequence.

The flash commands are not integrated into the ECU application, but into a small separate application, called the flash kernel. The flash kernel consists of the dSPACE Calibration and Bypassing Service and the custom-specific flash functions. The flash kernel is loaded by the flash tool to the ECU RAM, where it is activated.

Communication via dSPACE mailbox

Communication between the dSPACE ECU interface and the ECU is via a mailbox in the tool memory, called the dSPACE mailbox. Using several identifiers, the dSPACE Calibration and Bypassing Service checks if the mailbox configuration is valid (that is, if a host interface like the DCI-GSI2 is active at the moment). If the mailbox is valid, the dSPACE Calibration and Bypassing Service can execute a command each time the `dsecu_service_background` function is called. The service reads the relevant command ID and the required parameters from the mailbox, clears the command ID, and executes the corresponding command (for example, programming or erasing a memory segment). After the ECU has completed the command execution, the result parameters (for example, error

codes) are placed in the mailbox. Finally, the service signals the host interface that the command execution has finished.

Related topics

References

Configuration of ECU Flash Programming Features.....	128
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dSPACE Calibration and Bypassing Service API Functions

Introduction

Some functions of the dSPACE Calibration and Bypassing Service API must be called by the ECU application to initialize the service or to transfer data to or from the ECU.

Where to go from here

Information in this section

dsecu_service_init.....	64
To initialize the dSPACE Calibration and Bypassing Service.	
dsecu_service.....	64
To start the dSPACE Calibration and Bypassing Service foreground service.	
dsecu_service_state.....	66
To check the state of the specified service ID.	
dsecu_service_background.....	67
To start the dSPACE Calibration and Bypassing Service background service.	
dsecu_service_buffer_sync.....	68
To ensure that delayed data from the previous sampling step is recognized as old data.	
dsecuboot_check.....	69
To check at ECU boot time whether the calibration interface is requesting a flash operation.	

dsecu_service_init

Syntax

```
void dsecu_service_init (DSECU_UInt8 tool_nr)
```

Include file

dsecusvc.h

Purpose

To initialize the dSPACE Calibration and Bypassing Service.

Description

The `dsecu_service_init` function initializes the dSPACE Calibration and Bypassing Service and resets the internal variables and flags for one tool. The function must be called prior to any other function of the dSPACE Calibration and Bypassing Service. A typical location in the ECU code is where other initializations of the ECU are performed. You need to access the tool RAM for this, so the chip select must first be suitably programmed.

If the service is configured to use two instances, the `dsecu_service_init` function must be called twice, that is, with 0 and 1 as parameters. The parameter values are assigned to the tools via the addresses specified by `DSECU_DPMEM_OFFSETADDRx` in the configuration file. The addresses themselves are specified in the corresponding A2L file.

For further information, refer to [Integration of the dSPACE Calibration and Bypassing Service in the ECU Code](#) on page 27.

Parameters

tool_nr Specifies the number of the tool to be initialized. The number can be 0 or 1.

Return value

None

Related topics

References

dsecu_service.....	64
dsecu_service_background.....	67
dSPACE Calibration and Bypassing Service API Functions.....	63

dsecu_service

Syntax

```
int dsecu_service(DSECU_UInt16 SvcId)
```


Include file	<code>dsecusvc.h</code>										
Purpose	To start the dSPACE Calibration and Bypassing Service foreground service.										
Description	<p>The function is called to transfer data to or from the ECU. The function must be integrated in the ECU code only if data acquisition (DAQ) or bypassing is to be performed.</p> <p>For bypassing purposes, one or two function calls are necessary, depending on the bypass scenario:</p> <ul style="list-style-type: none"> ▪ Bypassing using two function calls One function call is needed at the beginning of the function or task to be bypassed (to read the arguments of the functions to be bypassed from the ECU). The other function call is needed at the end of the function or task to be bypassed (to write the function results calculated by the bypassing tool back to the ECU). ▪ Bypassing using one function call The function call is placed at the end of the function or task to be bypassed (to read the arguments of the functions to be bypassed by the tool, and to write the function results of the last sampling step calculated by the bypassing tool back to the ECU). <p>To perform data acquisition, only one function call placed at the end of the ECU task is needed.</p> <p>For further information, refer to Integration of the dSPACE Calibration and Bypassing Service in the ECU Code on page 27.</p>										
Parameters	SvcId Specifies the service ID in the range 1 ... 65535.										
Return value	<p>The function returns an error code. The error code covers the errors for both tools. For tool number 0, the error code is in the least significant byte (bits 0 ... 7), and for tool number 1, it is in bits 8 ... 15.</p> <p>The following error definitions can be used as flags, that is, they can be combined with a binary OR operation:</p> <table border="1"> <thead> <tr> <th>Predefined Symbol</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>DSECU_SVC_ACTIVE</td><td>This particular service is enabled. Data has been copied.</td></tr> <tr> <td>DSECU_SVC_INACTIVE</td><td>Either this particular service call, or all service calls, are disabled.</td></tr> <tr> <td>DSECU_FAILSAFE_DATA</td><td>No new bypassing data is available. Fail-safe data has been copied. The maximum number of failures is exceeded.</td></tr> <tr> <td>DSECU_NODATA_COPIED</td><td>No data has been copied. The maximum number of failures is exceeded.</td></tr> </tbody> </table>	Predefined Symbol	Meaning	DSECU_SVC_ACTIVE	This particular service is enabled. Data has been copied.	DSECU_SVC_INACTIVE	Either this particular service call, or all service calls, are disabled.	DSECU_FAILSAFE_DATA	No new bypassing data is available. Fail-safe data has been copied. The maximum number of failures is exceeded.	DSECU_NODATA_COPIED	No data has been copied. The maximum number of failures is exceeded.
Predefined Symbol	Meaning										
DSECU_SVC_ACTIVE	This particular service is enabled. Data has been copied.										
DSECU_SVC_INACTIVE	Either this particular service call, or all service calls, are disabled.										
DSECU_FAILSAFE_DATA	No new bypassing data is available. Fail-safe data has been copied. The maximum number of failures is exceeded.										
DSECU_NODATA_COPIED	No data has been copied. The maximum number of failures is exceeded.										

Predefined Symbol	Meaning
DSECU_NODATA_AVAILABLE	No data has been copied. No bypassing data is available.
DSECU_OLD_DATA	No new bypassing data is available. Old data has been copied.
DSECU_SVC_INVALID	The ID used in the service call is invalid. Maybe the maximum number of service IDs has to be increased (refer to DSECU_MAX_SERVICE_COUNT on page 127).

Related topics

References

dsecu_service_background	67
dsecu_service_init	64
dsecu_service_state	66
dSPACE Calibration and Bypassing Service API Functions	63

dsecu_service_state

Syntax

```
int dsecu_service_state(DSECU_UInt16 SvcId)
```

Include file

dsecusvc.h

Purpose

To check the state of the specified service ID.

Description

This function can be used to determine the state of a given service ID. If the service ID has not been enabled by the tool, **DSECU_SVC_INACTIVE** is returned. When it has been enabled, **DSECU_SVC_ACTIVE** is returned, and subsequent calls to the **dsecu_service** function will trigger the tool.

Parameters

SvcId Specifies the service ID in the range 1 ... 65535.

Return value

The function returns an error code that covers the errors for both tools. For tool number 0, the error code is in the least significant byte (bits 0 ... 7), and for tool number 1, it is in bits 8 ... 15.

The following codes can be returned by the function:

Predefined Symbol	Meaning
DSECU_SVC_ACTIVE	This particular service is enabled.
DSECU_SVC_INACTIVE	Either this particular service call, or all service calls, are disabled.

Predefined Symbol	Meaning
DSECU_SVC_INVALID	The ID used in the service call is invalid. Maybe the maximum number of service IDs has to be increased (refer to DSECU_MAX_SERVICE_COUNT on page 127).

Related topics

References

dsecu_service	64
dSPACE Calibration and Bypassing Service API Functions	63

dsecu_service_background

Syntax

```
int dsecu_service_background(void)
```

Include file

dsecusvc.h

Purpose

To start the dSPACE Calibration and Bypassing Service background service.

Description

The function must always be integrated into the ECU code. It must be called repeatedly in the background loop of the ECU, or in the task with the largest time interval if no background loop is available. The function checks whether the tool is/the tools are connected to the ECU, and whether it is/they are working correctly.

For further information, refer to [Integration of the dSPACE Calibration and Bypassing Service in the ECU Code](#) on page 27.

Parameters

None

Return value

The function returns an error code. The error code covers the errors for both tools. For tool number 0, the error code is in the least significant byte (bits 0 ... 7). Bits 8 ... 15 are used for the error code for tool number 1.

The following error definitions can be used as flags, that is, they can be combined with a binary OR operation:

Predefined Symbol	Meaning
DSECU_SERVICES_ENABLED	The tool is working correctly. Data is transmitted.
DSECU_SERVICES_DISABLED	The tool is not detected as alive. No data is transmitted.

Related topics

References

dsecu_service.....	64
dsecu_service_init.....	64
dSPACE Calibration and Bypassing Service API Functions.....	63

dsecu_service_buffer_sync

Syntax

```
int dsecu_service_buffer_sync(DSECU_UInt16 SvcId)
```

Include file

```
dsecusvc.h
```

Purpose

To ensure that delayed data from the previous sampling step is marked as old data.

Note

As of RTI Bypass Blockset 2.1, the `dsecu_service_buffer_sync` function is no longer required. The buffer synchronization settings can be set conveniently in the blockset, provided the `dsecu_buffer_sync` function is not used in the ECU code.

Nevertheless, the `dsecu_service_buffer_sync` function is still provided by the dSPACE Calibration and Bypassing Service for compatibility reasons.

Description

The dSPACE Calibration and Bypassing Service provides the `dsecu_service_buffer_sync` function, which avoids delayed results of the previous sampling step being erroneously interpreted as results of the current sampling step. `dsecu_service_buffer_sync` recognizes results that are written back to the ECU after the ECU waiting time (specified in the wait mechanism) has expired as old data of the previous sampling step. Delayed results from the previous sampling step are marked as old results by setting old buffer equal to current buffer (see [Flowcharts of extended bypassing mechanisms](#) on page 45).

The `dsecu_service_buffer_sync` function must be integrated in the ECU code only if bypassing is performed using two `dsecu_service` function calls, and if the wait mechanism is enabled. It must be called before the `dsecu_service` function at the beginning of the function or task to be bypassed (which reads the arguments of the functions to be bypassed to the tool). The argument of the `dsecu_service_buffer_sync` function must be

identical to that of the `dsecu_service` function at the end of the function or task to be bypassed (which writes the function results calculated by the bypassing tool back to the ECU).

Note

`dsecu_service_buffer_sync` applies to the data written from the RCP system to the ECU. The function is active only if the wait mechanism is enabled.

Parameters

SvcId Specifies the service ID in the range 1 ... 65535.

Return value

The function returns an error code. The error code covers the errors for both tools. For tool number 0, the error code is in the least significant byte (bits 0 ... 7), and tool number 1, it is in bits 8 ... 15.

The following error definitions can be used as flags, that is, they can be combined with a binary OR operation:

Predefined Symbol	Meaning
DSECU_SVC_ACTIVE	This particular service is enabled.
DSECU_SVC_INACTIVE	Either this particular service call, or all service calls, are disabled.
DSECU_SVC_INVALID	The ID used in the service call is invalid. Maybe the maximum number of service IDs has to be increased (refer to DSECU_MAX_SERVICE_COUNT on page 127).

Example

```
void task(void)
{
    dsecu_service_buffer_sync(3);
    dsecu_service(2);
    ...
    dsecu_service(3);
}
```

Related topics

References

[dSPACE Calibration and Bypassing Service API Functions..... 63](#)

dsecuboot_check

Syntax

```
void dsecuboot_check(void)
```

Include file	<code>dsECUboot.h</code>
Purpose	To check at ECU boot time whether a flashing request is pending and to start the flash kernel from your ECU application.
Description	<p>It is recommended to call this function as soon as possible in the ECU software, so that it is possible to flash the ECU even if a part of the ECU software is erroneous or damaged. You need to access the tool RAM for this, so the chip select must first be suitably configured. <code>dsecuboot_check</code> must be called before the <code>dsecu_service_init</code> function.</p> <p>This function must be called only if ECU flash programming is to be performed with the dSPACE Calibration and Bypassing Service.</p>
Parameters	None
Return value	None
Related topics	<div>References dSPACE Calibration and Bypassing Service API Functions..... 63</div>

dSPACE Calibration and Bypassing Service Custom API Functions

Introduction The dSPACE Calibration and Bypassing Service Custom API provides several functions.

Where to go from here **Information in this section**

Basics on the dSPACE Calibration and Bypassing Service Custom API.....	72
The dSPACE Calibration and Bypassing Service provides several functions for ECU flash programming, calibration, page switching, and interrupt handling. Since these features depend on customer-specific solutions,	

the appropriate standard dSPACE Calibration and Bypassing Service API functions must be adapted.

dsecu_custom_flash_program_start.....	73
To start the ECU flash programming.	
dsecu_custom_flash_erase_range.....	74
To erase a range in the ECU flash memory.	
dsecu_custom_flash_program_range.....	75
To program a range of the ECU flash memory.	
dsecu_custom_flash_decode.....	76
To decrypt and decode data that are to be flashed.	
dsecu_custom_flash_finish.....	77
To finish the ECU flash programming.	
dsecu_custom_command.....	78
To add custom commands to the dSPACE Calibration and Bypassing Service.	
dsecu_custom_copy.....	79
To read and write memory from/to the ECU.	
dsecu_custom_cal_page_switch.....	80
To switch the calibration page on the ECU.	
dsecu_custom_cal_absolute_access.....	81
To check the access rights in the absolute address mode.	
dsecu_custom_cal_paged_access.....	82
To check the access rights in the paged address mode.	
dsecu_custom_cal_paged_access_finished.....	83
To finish the paged address mode.	
dsecu_custom_trigger_hw_int.....	84
To trigger hardware interrupts on the calibration or bypass interface.	
dsecu_custom_timeout_timestamp_get.....	85
To calculate a custom time stamp.	
dsecu_custom_timeout_pending_check.....	86
To check periodically whether a timeout occurred.	

Basics on the dSPACE Calibration and Bypassing Service Custom API

Introduction

The dSPACE Calibration and Bypassing Service Custom API provides several functions.

Basics

The dSPACE Calibration and Bypassing Service provides several functions for ECU flash programming, calibration, page switching, and interrupt handling. Since these features depend on customer-specific solutions, the appropriate standard dSPACE Calibration and Bypassing Service API functions must be adapted. In addition, it is possible to extend the dSPACE Calibration and Bypassing Service by custom commands (included in a dSPACE engineering project).

To implement the standard API functions custom-specifically, the functions must be modified to map the dSPACE Calibration and Bypassing Service API to customer-specific functions.

Related topics**Basics**

[Integration of the dSPACE Calibration and Bypassing Service in the ECU Code..... 27](#)

dsecu_custom_flash_program_start

Syntax

```
DSECU_UInt32 dsecu_custom_flash_program_start(void)
```

Include file

dsECUcustom.h

Purpose

To start the programming of the ECU flash memory.

Description

This function requests permission to start a flash operation. The command is always executed before the first erase or program operation is started. The ECU may refuse a request if the current ECU state does not allow an update, for example, if the engine is still running. In that case, the function returns **DSECU_UPDATE_DENIED**. Otherwise, if permission is granted, the ECU may also go to a different mode in which the standard ECU application is no longer running, for example, by performing a system reset and starting the ECU's boot code. Then the function returns **DSECU_NO_ERROR**.

Parameters

None

Return value The function returns the following error code:

Predefined Symbol	Meaning
DSECU_NO_ERROR	The function has been executed successfully.
DSECU_UPDATE_DENIED	The update has been refused at the current ECU state.

Related topics

References

dsecu_custom_flash_decode.....	76
dsecu_custom_flash_erase_range.....	74
dsecu_custom_flash_finish.....	77
dsecu_custom_flash_program_range.....	75
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_flash_erase_range

Syntax

```
DSECU_UInt32 dsecu_custom_flash_erase_range(
    DSECU_UInt32 address,
    DSECU_UInt32 length)
```

Include file

dsECUcustom.h

Purpose

To erase a range in the ECU flash memory.

Description

This function returns **DSECU_NO_ERROR** if the range has been erased successfully. Otherwise, it returns **DSECU_ERASE_FAILED**.

Parameters

address Specifies the first address to be erased.
length Specifies the length of the block to be erased.

Return value The function returns the following error code:

Predefined Symbol	Meaning
DSECU_NO_ERROR	The range has been erased successfully.
DSECU_ERASE_FAILED	The range has not been erased.
DSECU_BLANKCHECK_FAILED	The range has been programmed, but verifying the erased range has failed.

Related topics

References

dsecu_custom_flash_decode.....	76
dsecu_custom_flash_finish.....	77
dsecu_custom_flash_program_range.....	75
dsecu_custom_flash_program_start.....	73
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_flash_program_range

Syntax

```
DSECU_UInt32 dsecu_custom_flash_program_range(
    DSECU_UInt32 address,
    DSECU_UInt32 length,
    void* data)
```

Include file

dsECUcustom.h

Purpose

To program a range of the ECU flash memory.

Description

Programming a range of the ECU flash memory includes the verification of the programmed range. The function returns **DSECU_NO_ERROR** if the range has been programmed successfully. If the range has been programmed but the check on the programmed range has failed, the function returns **DSECU_VERIFY_FAILED**. Otherwise, it returns **DSECU_PROGRAM_FAILED**.

Parameters

address Specifies the first address to be programmed.

length Specifies the length of the block to be programmed.

data Pointer to the data.

Return value

The function returns the following error code:

Predefined Symbol	Meaning
DSECU_NO_ERROR	The range has been programmed successfully.
DSECU_PROGRAM_FAILED	The range has not been programmed.
DSECU_VERIFY_FAILED	The range has been programmed, but verifying the programmed range has failed.

Related topics

References

dsecu_custom_flash_decode.....	76
dsecu_custom_flash_erase_range.....	74
dsecu_custom_flash_finish.....	77
dsecu_custom_flash_program_start.....	73
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_flash_decode

Syntax

```
DSECU_UInt32 dsecu_custom_flash_decode(
    DSECU_UInt8 compression_method,
    DSECU_UInt8 encryption_method,
    void* input_data,
    DSECU_UInt32 input_length,
    void** output_data,
    DSECU_UInt32* output_length)
```

Include file

dsECUcustom.h

Purpose

To decrypt and decode data that is to be flashed.

Description

This function decrypts and decodes the given data and returns the decrypted/decoded data. The size of the returned data must be stored in the **output_length** parameter. The function returns **DSECU_NO_ERROR** if the data has been successfully decrypted and decoded. Otherwise, it returns **DSECU_METHOD_NOT_SUPPORTED**.

dsecu_custom_flash_decode is an optional function which has to be filled only if encrypted and compressed data is to be flashed.

Note

Currently, the encryption and compression feature is not provided by the dSPACE ECU Flash Programming Tool.

Parameters	compression_method Specifies the compression method. encryption_method Specifies the encryption method. input_data Pointer to the input data. input_length Specifies the length of the input buffer. output_data Pointer to the output data. output_length Specifies the size of the decrypted and decompressed data.
-------------------	--

Return value The function returns the following error code:

Predefined Symbol	Meaning
DSECU_NO_ERROR	The data has been decrypted and decoded successfully.
DSECU_METHOD_NOT_SUPPORTED	The decryption and decompression of the data has failed.

Related topics

References

dsecu_custom_flash_erase_range.....	74
dsecu_custom_flash_finish.....	77
dsecu_custom_flash_program_range.....	75
dsecu_custom_flash_program_start.....	73
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_flash_finish

Syntax `DSECU_UInt32 dsecu_custom_flash_finish(void)`

Include file `dsECUcustom.h`

Purpose To finish the programming of the ECU flash memory.

Description This function is called at the end, when the update operation has finished. The ECU application can take appropriate measures, for example, to lock the flash memory.

Parameters None

Return value	DSECU_NO_ERROR The function has been executed successfully.
---------------------	--

Related topics**References**

dsecu_custom_flash_decode.....	76
dsecu_custom_flash_erase_range.....	74
dsecu_custom_flash_program_range.....	75
dsecu_custom_flash_program_start.....	73
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_command

Syntax

```
DSECU_UInt32 dsecu_custom_command(
    DSECU_UInt32 command_id,
    DSECU_UInt32* data_length,
    DSECU_UInt8* data,
    DSECU_UInt32 max_return_data)
```

Include file

dsECUcustom.h

Purpose

To call customer-specific commands in the dSPACE Calibration and Bypassing Service.

Description

This function is called to add a custom command to the dSPACE Calibration and Bypassing Service. The custom function can use the **command_id** to select the execution of a specific command. Further parameters can be specified by the **data** and **data_length** parameters. The former holds the data, the latter specifies its number of bytes. If the function is to return data, the **data_length** parameter specifies the number of bytes that are to be returned to the caller, and the return data is stored in the **data** parameter again. The function should set the **data_length** to 0, if no data is to be returned. If the return value of the function does not equal **DSECU_NO_ERROR**, a corresponding error code must be returned to the host. The **max_return_data** parameter specifies the maximum number of bytes which may be returned in the **data** parameter. When this function is called, **data_length** parameter must be between 0 and **max_return_data**.

Note

A custom command can only be implemented as part of a dSPACE engineering service, because it requires modifications to the DCI-GSI2's firmware.

Parameters	command_id	Specifies the custom command ID.
	data_length	Specifies the number of bytes of data .
	data	Specifies the data or the return data.
	max_return_data	Specifies the maximum number of bytes that may be returned.

Return value	DSECUCUSTOM_NO_ERROR	The function has been executed successfully.
---------------------	-----------------------------	--

Related topics**References**

[dSPACE Calibration and Bypassing Service Custom API Functions..... 71](#)

dsecu_custom_copy

Syntax	<pre>DSECUCUSTOM_UINT32 dsecu_custom_copy(DSECUCUSTOM_UINT32 dest_address, DSECUCUSTOM_UINT32 source_address, DSECUCUSTOM_UINT32 length)</pre>
---------------	---

Include file	<code>dsECUCustom.h</code>
---------------------	----------------------------

Purpose	To read and write memory from/to the ECU.
----------------	---

Description	This function copies the specified number of bytes from the source address to the destination address.
--------------------	--

dsecu_custom_copy is used for uploading the contents of the flash memory during flash programming and for accessing the ECU memory if the `DSECUCUSTOM_CAL_MEMORY_COPY_METHOD` configuration option is set to `DSECUCUSTOM_MEMCPY_CUSTOM`. By default, this function contains a simple call to the `memcpy()` function.

Parameters	dest_address	Specifies the destination address the copied data is to be written to.
	source_address	Specifies the source address of the data to be copied.
	length	Specifies the number of bytes to be copied.

Return value	DSECU_NO_ERROR	The memory has been copied successfully.
---------------------	-----------------------	--

Related topics**References**

DSECU_CAL_MEMORY_COPY_METHOD.....	108
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_cal_page_switch

Syntax

```
DSECU_UInt32 dsecu_custom_cal_page_switch(
    DSECU_UInt8 page_number)
```

Include file

dsECUcustom.h

Purpose

To switch the calibration page on the ECU.

Description

This function switches to the calibration page specified by the **page_number** parameter. The function returns **DSECU_NO_ERROR** if the page switch has been successful. If the specified page does not exist, it returns **DSECU_ILLEGAL_PAGE**.

Parameters

page_number Specifies the number of the page that is to be activated. The page number must be in the range 0 ... (**DSECU_CAL_NUMBER_OF_PAGES** - 1).

Return value

The function returns the following error code:

Predefined Symbol	Meaning
DSECU_NO_ERROR	The page has been switched successfully.
DSECU_ILLEGAL_PAGE	The page does not exist.

Related topics

References

dsecu_custom_cal_absolute_access.....	81
dsecu_custom_cal_paged_access.....	82
dsecu_custom_cal_paged_access_finished.....	83
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_cal_absolute_access

Syntax

```
DSECU_UInt32 dsecu_custom_cal_absolute_access(
    DSECU_UInt32 address,
    DSECU_UInt32 size,
    DSECU_UInt8 write_access)
```

Include file

dsECUcustom.h

Purpose

To check permission for access in the absolute address mode.

Description

This function checks whether access to the memory block which is specified by **address** and **size** is allowed in the absolute address mode. (In absolute address mode, a parameter is written directly to the specified address.) Memory blocks can be handled as read-only via the **write_access** parameter. If write access is granted, the memory block is also readable.

Access to the specified address is prohibited unless the **DSECU_NO_ERROR** error code is returned.

Parameters

address Specifies the address which is to be checked.

size Specifies the size of the block that is to be accessed.

write_access Specifies whether write access is requested (1) or not (0).

Return value

DSECU_NO_ERROR Access has been granted to the memory block.

Related topics

References

dsecu_custom_cal_page_switch.....	80
dsecu_custom_cal_paged_access.....	82
dsecu_custom_cal_paged_access_finished.....	83
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_cal_paged_access

Syntax

```
DSECUCustom dsecu_custom_cal_paged_access(
    DSECUCustom page_number,
    DSECUCustom* address,
    DSECUCustom size,
    DSECUCustom write_access)
```

Include file

dsECUCustom.h

Purpose

To check permission for access in the paged address mode.

Description

This function translates a paged address to an absolute address, and then checks whether access for a memory block is allowed in paged address mode. (In absolute address mode, a parameter is written directly to the specified address. This addressing mode must be used if data has to be written to a memory region that is not paged.) The paged address is given by the **address** of the parameter and the page specified by the **page_number**. The memory block is specified by the **address** and the **size**. The absolute address must be returned via **address**.

Note

The **address** parameter does not contain the address itself, but a pointer to it.

Memory blocks can be handled as read-only via the **write_access** parameter. If write access is granted, the memory block is also readable.

Access to the specified address is prohibited unless the **DSECUCustom_NO_ERROR** error code is returned.

Parameters	<p>page_number Specifies the number of the page that is to be accessed. The page number must be in the range 0 ... (DSECU_CAL_NUMBER_OF_PAGES - 1).</p> <p>address Pointer to the address that is to be accessed.</p> <p>size Specifies the number of bytes that are to be accessed at the given address.</p> <p>write_access Specifies whether write access to the page is requested (1) or not (0).</p>								
Return value	DSECU_NO_ERROR Access has been granted to the memory block.								
Related topics	<p>References</p> <table> <tr> <td>dsecu_custom_cal_absolute_access.....</td><td>81</td></tr> <tr> <td>dsecu_custom_cal_page_switch.....</td><td>80</td></tr> <tr> <td>dsecu_custom_cal_paged_access_finished.....</td><td>83</td></tr> <tr> <td>dSPACE Calibration and Bypassing Service Custom API Functions.....</td><td>71</td></tr> </table>	dsecu_custom_cal_absolute_access.....	81	dsecu_custom_cal_page_switch.....	80	dsecu_custom_cal_paged_access_finished.....	83	dSPACE Calibration and Bypassing Service Custom API Functions.....	71
dsecu_custom_cal_absolute_access.....	81								
dsecu_custom_cal_page_switch.....	80								
dsecu_custom_cal_paged_access_finished.....	83								
dSPACE Calibration and Bypassing Service Custom API Functions.....	71								

dsecu_custom_cal_paged_access_finished

Syntax	<code>DSECU_UInt32 dsecu_custom_cal_paged_access_finished(void)</code>
Include file	<code>dsECUcustom.h</code>
Purpose	To finish the paged address mode.
Description	This function is called after access to a paged memory block is completed. For example, if a chip select must be reconfigured on the ECU for accessing a certain page, this is done in the <code>dsecu_custom_cal_paged_access</code> function. The <code>dsecu_custom_cal_paged_access_finished</code> function is used to revoke it.
Parameters	None
Return value	DSECU_NO_ERROR The function has been executed successfully.

Related topics

References

dsecu_custom_cal_absolute_access.....	81
dsecu_custom_cal_page_switch.....	80
dsecu_custom_cal_paged_access.....	82
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_trigger_hw_int

Syntax

```
void dsecu_custom_trigger_hw_int(
    int tool_nr,
    DSECU_UInt16 SvcId)
```

Include file

dsECUcustom.h

Purpose

To trigger a hardware interrupt on the bypass or ECU interface.

Description

For every call of `dsecu_service`, a hardware interrupt is triggered on the ECU interface or the RCP system. A subinterrupt mechanism is used to manage different interrupts for different service IDs. The `dsecu_custom_trigger_hw_int` function only triggers the hardware interrupts. The subinterrupts are handled in other functions.

The trigger mechanism depends on the ECU hardware. Here are some examples of hardware interrupt triggering:

- Toggling a digital I/O pin which is connected to the calibration or bypassing interface
- Configuring a watchpoint (for example, on the MPC56x from Freescale, formerly Motorola) and writing to a specific memory location
- On some processors (for example, the NEC V85x) the trigger cannot be configured by the processor itself, but must be set by the calibration or bypassing interface. In these cases, the DCI Configuration Tool or the RTI Bypass Blockset is used for configuration.

Parameters

tool_nr Specifies the number of the tool (0 or 1).

SvcId Specifies the service ID of the `dsecu_service` call.

Return value

None

Related topics**References**

DSECU_CUSTOM_HW_TRIGGER.....	124
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_timeout_timestamp_get

Syntax

```
DSECU_TIMEOUT_TIMESTAMP_TYPE dsecu_custom_timeout_timestamp_get(
    DSECU_UInt16 SvcID,
    DSECU_Int32 TimeOut)
```

Include file

dsECUcustom.h

Purpose

To calculate a custom time stamp.

Description

This function calculates the current time stamp at the beginning of a DAQ list, before capturing data from the ECU is started.

dsecu_custom_timeout_timestamp_get must be called before an ECU read waiting sequence.

The returned time stamp value will be used by **dsecu_custom_timeout_pending_check** for timeout calculation.

The **dsecu_custom_timeout_timestamp_get** function is required only if **DSECU_TIMEOUT_CUSTOM_FUNCTION DSECU_ENABLED** is used.

Parameters

SvcId Specifies the service ID of the **dsecu_service** call.

TimeOut Specifies the timeout value in 10 µs.

Return value

TimeStamp Returns the calculated custom time stamp value.

Related topics**References**

dsecu_custom_timeout_pending_check.....	86
DSECU_TIMEOUT_CUSTOM_FUNCTION.....	119
DSECU_TIMEOUT_TIMESTAMP_TYPE.....	121
dSPACE Calibration and Bypassing Service Custom API Functions.....	71

dsecu_custom_timeout_pending_check

Syntax

```
DSECU_UInt32 dsecu_custom_timeout_pending_check(  
    DSECU_UInt16 SvcID,  
    DSECU_Int32 TimeOut,  
    DSECU_TIMEOUT_TIMESTAMP_TYPE TimeStamp)
```

Include file

dsECUcustom.h

Purpose

To check periodically whether a timeout occurred.

Description

This function checks whether the time stamp returned by `dsecu_custom_timeout_timestamp_get` caused a timeout.

`dsecu_custom_timeout_pending_check` is called periodically during the ECU read process until all the data associated with the DAQ list is captured, or a timeout is indicated by this function.

The `dsecu_custom_timeout_pending_check` function is required only if `DSECU_TIMEOUT_CUSTOM_FUNCTION DSECU_ENABLED` is used.

Parameters

SvcId Specifies the service ID of the `dsecu_service` call.

TimeOut Specifies the timeout value in 10 μ s.

TimeStamp Specifies the custom time stamp returned by `dsecu_custom_timeout_timestamp_get`.

Return value

The function returns the TimeoutPending value as follows:

TimeoutPending Value	Meaning
0	A timeout occurred.
1	The timeout is pending.

Related topics**References**

dsecu_custom_timeout_timestamp_get	85
DSECU_TIMEOUT_CUSTOM_FUNCTION	119
DSECU_TIMEOUT_TIMESTAMP_TYPE	121
dSPACE Calibration and Bypassing Service Custom API Functions	71

dSPACE Calibration and Bypassing Service Configuration Options

Introduction

The dSPACE Calibration and Bypassing Service provides mechanisms for calibration, data acquisition, bypassing, subinterrupt handling, and ECU flash programming. Most of these features are configurable.

Where to go from here

Information in this section

Basics on Configuring Features of the dSPACE Calibration and Bypassing Service.....	88
The dSPACE Calibration and Bypassing Service provides options to configure its layers and features.	
Configuration of the Enabling/Disabling Features.....	90
The dSPACE Calibration and Bypassing Service provides several options for enabling/disabling the main features of the service.	
Configuration of General Features.....	96
The dSPACE Calibration and Bypassing Service provides several options for specifying general configuration options for the dSPACE Calibration and Bypassing Service.	
Configuration of Calibration Features.....	106
The dSPACE Calibration and Bypassing Service provides several options for configuring calibration features.	
Configuration of DAQ and Bypassing Features.....	110
The dSPACE Calibration and Bypassing Service provides several options for configuring data acquisition (DAQ) and bypassing features.	
Configuration of ECU Flash Programming Features.....	128
The dSPACE Calibration and Bypassing Service provides several options for configuring ECU flash programming features.	

Basics on Configuring Features of the dSPACE Calibration and Bypassing Service

Introduction

The dSPACE Calibration and Bypassing Service provides options to configure its layers and features.

Basics on Configuration Options

Introduction

The features of the dSPACE Calibration and Bypassing Service can be configured. Configuration is done via the dSPACE Calibration and Bypassing Service configuration files `dsECUcfg.h` (for DAQ and bypassing) and `dsECUbootcfg.h` (ECU flash programming), by means of conditional compilation with preprocessor instructions (`#defines`).

Basic information on configuration options

The dSPACE Calibration and Bypassing Service supports bypassing, data acquisition and ECU flash programming tools in parallel. For the DCI-GSI2, it is sufficient to enable one service instance to perform bypassing, data acquisition or ECU flash programming in parallel.

Some of the configuration options apply to both tools, which means that the two service instances can be configured together (for example, the support of 32-bit data types that can only be enabled globally). Other configuration options must be set individually for each tool (for example, the location of the service tables). Some preprocessor instructions are defined globally and must be unique. In these cases, the name of each configuration option ends with the number of the service instance. The configuration options of the first service instance have the suffix "0", and the configuration options of the second service instance are suffixed by "1".

Note

If only one tool is used at a time, the configuration options of the first service instance are used, and the configuration options with suffix 1 are ignored.

In general representation, the names of the affected options end in "x".

Note

Some customizing modifications of the dSPACE Calibration and Bypassing Service require corresponding modifications in the associated `IF_DATA` element of the ECU's A2L file.

Related topics

References

Configuration of Calibration Features.....	106
Configuration of DAQ and Bypassing Features.....	110
Configuration of ECU Flash Programming Features.....	128
Configuration of General Features.....	96
Configuration of the Enabling/Disabling Features.....	90

Configuration of the Enabling/Disabling Features

Introduction

The dSPACE Calibration and Bypassing Service provides configuration options that enable/disable the main features of the service.

Disabling parts of the service reduces the consumption of ECU memory, and may therefore improve the performance of the ECU application.

Where to go from here

Information in this section

DSECU_SERVICE_REMOVED.....	90
To enable or disable the dSPACE Calibration and Bypassing Service completely.	
DSECU_DAQ_BYPASS_SERVICE.....	91
To enable or disable data acquisition (DAQ) and bypassing.	
DSECU_BYPASS.....	92
To enable or disable bypassing.	
DSECU_CALIBRATION_COMMANDS.....	92
To enable or disable calibration commands.	
DSECU_CUSTOM_COMMANDS.....	93
To enable or disable customer-specific commands.	
DSECU_FLASH_COMMANDS.....	94
To enable or disable ECU flash programming commands.	

DSECU_SERVICE_REMOVED

Syntax

```
#define DSECU_SERVICE_REMOVED
```

Purpose

To enable or disable completely the dSPACE Calibration and Bypassing Service.

Description

DSECU_SERVICE_REMOVED disables the compilation of the dSPACE Calibration and Bypassing Service. A common way to disable the dSPACE Calibration and Bypassing Service is to pass this option to the compiler as a parameter.

Parameters

None

Example

```
#define DSECU_SERVICE_REMOVED
```

Related topics**Basics**

Basics on Configuration Options..... 88

DSECU_DAQ_BYPASS_SERVICE

Syntax

```
#define DSECU_DAQ_BYPASS_SERVICE
```

Include file

dsecucfg.h

Purpose

To enable or disable data acquisition (DAQ) and bypassing.

Description

During data acquisition and bypassing, a mechanism copies data to or from the tool RAM. If DSECU_DAQ_BYPASS_SERVICE is disabled the mechanism is disabled, and data acquisition and bypassing are not available. Other features, like ECU flash programming, are not affected.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables DAQ and bypassing.
DSECU_DISABLED	Disables DAQ and bypassing.

Example

```
#define DSECU_DAQ_BYPASS_SERVICE DSECU_ENABLED
```

Related topics**References**

DSECU_BYPASS..... 92

DSECU_BYPASS

Syntax

```
#define DSECU_BYPASS
```

Include file

dsecucfg.h

Purpose

To enable or disable bypassing.

Description

During bypassing, a mechanism copies data from the tool RAM to the internal RAM of the ECU processor. If DSECU_BYPASS is disabled, this mechanism is disabled, and bypassing is not available. Copying data in the other direction is not affected, so that data acquisition is still possible.

Using this configuration option makes sense only if DAQ and bypassing are enabled by DSECU_DAQ_BYPASS_SERVICE.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables bypassing.
DSECU_DISABLED	Disables bypassing.

Example

```
#define DSECU_BYPASS DSECU_DISABLED
```

Related topics

References

DSECU_DAQ_BYPASS_SERVICE..... 91

DSECU_CALIBRATION_COMMANDS

Syntax

```
#define DSECU_CALIBRATION_COMMANDS
```

Include file

dsecucfg.h

Purpose To enable or disable calibration commands.

Description Other commands (for example, ECU flash programming commands) are not affected by this option.

Note

For some DCI-GSI2 processor families (e.g., the Freescale MPC55xx family), calibration is done using the dSPACE Calibration and Bypassing Service. In these cases, you must enable the calibration commands.

Parameters The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables ECU calibration commands.
DSECU_DISABLED	Disables ECU calibration commands.

Example

```
#define DSECU_CALIBRATION_COMMANDS DSECU_DISABLED
```

Related topics

References

Configuration of Calibration Features..... 106

DSECU_CUSTOM_COMMANDS

Syntax

```
#define DSECU_CUSTOM_COMMANDS
```

Include file dsecucfg.h

Purpose To enable or disable customer-specific commands.

Description

Other commands (for example, ECU flash programming commands) are not affected by this option.

Note

- For the DCI-GSI2, the use of customer-specific commands requires customer-specific firmware on the DCI-GSI2. The `DSECU_CUSTOM_COMMANDS` define must therefore be set to `DSECU_DISABLED` in most cases.
- For a RapidPro system used as a stand-alone prototyping ECU, the required custom commands and configuration settings are fixed and already integrated in the dSPACE Calibration and Bypassing Service.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_ENABLED</code>	Enables ECU customer-specific commands.
<code>DSECU_DISABLED</code>	Disables ECU customer-specific commands.

Example

```
#define DSECU_CUSTOM_COMMANDS DSECU_DISABLED
```

Related topics**Basics**

[Basics on Configuration Options..... 88](#)

DSECU_FLASH_COMMANDS

Syntax

```
#define DSECU_FLASH_COMMANDS
```

Include file

`dsecucfg.h`

Purpose

To enable or disable ECU flash programming commands.

Description

Other commands (for example, the custom command) are not affected by this option.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables ECU flash programming commands.
DSECU_DISABLED	Disables ECU flash programming commands.

Example

```
#define DSECU_FLASH_COMMANDS DSECU_DISABLED
```

Related topics

References

Configuration of ECU Flash Programming Features..... 128

Configuration of General Features

Introduction

Several defines are provided to specify general configuration options for the dSPACE Calibration and Bypassing Service.

Where to go from here

Information in this section

DSECU_NUMBER_TOOLS.....	96
To specify the number of DAQ and bypassing tools that can be used at the same time.	
DSECU_INT_STATUS_VAR_DECL.....	97
To declare a status variable for target processor interrupts.	
DSECU_INT_SAVE_AND_DISABLE.....	98
To disable the interrupts on the target processor.	
DSECU_INT_RESTORE.....	99
To re-enable the interrupts on the target processor.	
DSECU_POINTER_QUALIFIER.....	100
To specify a prefix for pointer declarations.	
DSECU_INLINE.....	101
To specify a keyword for inline functions.	
DSECU_DPMEM_OFFSETADDRx.....	101
To specify the start address of the tool RAM.	
DSECU_DPMEM_SIZEEx.....	102
To specify the size of the tool RAM.	
DSECU_POD_TYPEEx.....	103
To specify the type of the POD used to adapt the target.	
DSECU_PROCESSOR_TYPE.....	104
To specify the byte order of the ECU processor.	
DSECU_BYTE_WRITE_SUPPORTED.....	105
To specify whether byte-wise write access to the tool RAM is supported.	

DSECU_NUMBER_TOOLS

Syntax

```
#define DSECU_NUMBER_TOOLS
```


Include file	<ul style="list-style-type: none"> ▪ dsecucfg.h ▪ dsecubootcfg.h
Purpose	To specify the number of tools for DAQ, bypassing, or ECU flash programming supported by the dSPACE Calibration and Bypassing Service at the same time.
Description	<p>For example, you can use either one calibration tool, or one ECU flashing tool, or one calibration and one bypassing tool at the same time.</p> <p>For the DCI-GSI2, it is sufficient to enable one tool. The DCI-GSI2 translates the accesses of several tools into service accesses while using only one service instance.</p>
Parameters	<p>number of tools Specifies the number of service instances in the range 1 ... 2.</p>
Example	<pre>#define DSECU_NUMBER_TOOLS 2</pre>
Related topics	<p>Basics</p> <p>Basics on Configuration Options..... 88</p>

DSECU_INT_STATUS_VAR_DECL

Syntax	<pre>#define DSECU_INT_STATUS_VAR_DECL(status)</pre>
Include file	dsecucfg.h
Purpose	To declare a status variable for target processor interrupts.
Description	The DSECU_INT_SAVE_AND_DISABLE and DSECU_INT_RESTORE macros require the status parameter. This is a variable that can be used to store the current interrupt status while interrupts are disabled.

If you use the variable within at least one of the above macros, you must specify its declaration via the `DSECU_INT_STATUS_VAR_DECL` macro. In this macro, the variable must always be named `status`. The data type can be chosen as needed.

Tip

To prevent compiler warnings (e.g. 'The variable xxx was declared but never referenced'), you can leave the `DSECU_INT_STATUS_VAR_DECL` macro blank if no variable is needed to store the interrupt status.

Parameters

status Name of the interrupt status variable
 ... Declaration for the interrupt status variable

Example

- If an interrupt status variable is needed:

```
#define DSECU_INT_STATUS_VAR_DECL(status) unsigned int status
```

- If no interrupt status variable is needed:

```
#define DSECU_INT_STATUS_VAR_DECL(status)
```

Related topics

References

DSECU_INT_RESTORE.....	99
DSECU_INT_SAVE_AND_DISABLE.....	98

DSECU_INT_SAVE_AND_DISABLE

Syntax

```
#define DSECU_INT_SAVE_AND_DISABLE(status)
```

Include file

`dsecucfg.h`

Purpose

To disable the interrupts on the target processor and store their current statuses.

Description

A status can be stored in the variable specified by `status`, or on the stack as shown in the example below. If a variable is used, it is strongly recommended to use a local variable. Otherwise the status is overwritten if the `DSECU_INT_SAVE_AND_DISABLE` macro is called twice in succession without the `DSECU_INT_RESTORE` macro being called in between.

Disabling interrupts depends on the target processor and often on the compiler.

Parameters

status Can be used to store the interrupt status.

... Code to disable the interrupts.

Example

```
#define DSECU_INT_SAVE_AND_DISABLE(status) \
{ \
    /* Example for MC68336 */ \
    asm(" MOVE SR,-(SP) ; save interrupt status"); \
    asm(" ORI #$0700,SR ; disable interrupts"); \
}
```

Related topics

References

DSECU_INT_RESTORE.....99

DSECU_INT_STATUS_VAR_DECL.....97

DSECU_INT_RESTORE

Syntax

```
#define DSECU_INT_RESTORE(status)
```

Include file

dsecucfg.h

Purpose

To re-enable the interrupts on the target processor and restore the interrupt status.

Description

The status can be copied from the variable specified by **status** or from the stack as shown in the example below to the interrupt status register of the target processor. If a variable is used, it is strongly recommended to use a local variable. Otherwise the status is overwritten if **DSECU_INT_SAVE_AND_DISABLE** is called twice in succession without the **DSECU_INT_RESTORE** macro being called in between.

Disabling interrupts depends on the target processor and often on the compiler.

Parameters

status Can be used to store the interrupt status.

... Code to re-enable the interrupts.

Example

```
#define DSECU_INT_RESTORE(status) \
{ \
    /* Example for MC68336 */ \
    asm(" MOVE    SR,D0    ; move actual SR in D0"); \
    asm(" ANDI   #$FF,D0   ; mask current X,N,Z,V,C flags"); \
    asm(" MOVE   (SP)+,D1  ; move old status from stack to D1"); \
    asm(" OR     D0,D1     ; change old flags to current flags"); \
    asm(" MOVE.W D1,SR     ; restore interrupt status"); \
}
```

Related topics**References**

DSECU_INT_SAVE_AND_DISABLE.....	98
DSECU_INT_STATUS_VAR_DECL.....	97

DSECU_POINTER_QUALIFIER

Syntax

```
#define DSECU_POINTER_QUALIFIER
```

Include file

- dsecucfg.h
- dsecubootcfg.h

Purpose

To specify the kind of pointer that is used within the dSPACE Calibration and Bypassing Service.

Description

Some processors are restricted in addressing a large memory range. Sometimes an additional keyword (for example, "far") must be used in the source code to address a memory location.

Parameters

... Qualifier for pointers in the dSPACE Calibration and Bypassing Service.

Example

```
#define DSECU_POINTER_QUALIFIER far
```

Related topics**Basics**

Basics on Configuration Options.....	88
--------------------------------------	----

DSECU_INLINE

Syntax	<pre>#define DSECU_INLINE</pre>
Include file	dsecucfg.h
Purpose	To enable inlining for the dSPACE Calibration and Bypassing Service.
Description	Some functions of the dSPACE Calibration and Bypassing Service can be compiled as inline functions. If inlining is supported by the compiler, this define can be the same as the define which is set with inlining enabled.
Parameters	... Define which is set if the application is compiled with inlining.
Example	<pre>#define DSECU_INLINE inline</pre>
Related topics	Basics Basics on Configuration Options..... 88

DSECU_DPMEM_OFFSETADDRx

Syntax	<pre>#define DSECU_DPMEM_OFFSETADDR0 #define DSECU_DPMEM_OFFSETADDR1</pre>
Include file	dsecucfg.h
Purpose	To specify the start address of the memory that is used by the dSPACE Calibration and Bypassing Service.
Description	This define specifies the start address of the tool RAM, which can be accessed by the target processor and the tool.

If bypassing and data acquisition are done in parallel, this option must be set for each tool individually. The memory ranges of the two tools must not overlap. See also [DSECU_DPMEM_SIZE_x](#) on page 102.

If only one tool is used at a time, the `DSECU_DPMEM_OFFSETADDR0` define is used.

Note

Do not place the tool RAM area in a cached memory area of the microcontroller. Using a cached memory area can lead to loss or delay of interrupt triggers and can cause problems with delays in reading/writing memory contents.

Parameters	offset address Specifies the start address of the memory in the DPMEM used by the service.
-------------------	---

Example	<pre>#define DSECU_DPMEM_OFFSETADDR0 0x7FF80000</pre>
----------------	---

Related topics

References

[DSECU_DPMEM_SIZE_x](#)..... 102

DSECU_DPMEM_SIZE_x

Syntax	<pre>#define DSECU_DPMEM_SIZE0 #define DSECU_DPMEM_SIZE1</pre>
---------------	--

Include file	<code>dsecucfg.h</code>
---------------------	-------------------------

Purpose	To specify the size of the tool RAM that is used by the dSPACE Calibration and Bypassing Service.
----------------	---

Description	<p>This define specifies the length of the tool RAM which can be accessed by the target processor and the tool.</p> <p>If bypassing and data acquisition are done in parallel, this option must be set individually for each tool. The memory ranges of the two tools must not overlap. See also DSECU_DPMEM_OFFSETADDR_x on page 101.</p>
--------------------	--

If only one tool is used at a time, the `DSECU_DPMEM_SIZE0` define is used.

Note

Do not place the tool RAM area in a cached memory area of the microcontroller. Using a cached memory area can lead to loss or delay of interrupt triggers and can cause problems with delays in reading/writing memory contents.

Parameters	size of tool RAM Specifies the length in bytes of the memory that is used by the dSPACE Calibration and Bypassing Service.
-------------------	---

Example	<pre>#define DSECU_DPMEM_SIZE0 0x1000</pre>
----------------	--

Related topics	References DSECU_DPMEM_OFFSETADDRx..... 101
-----------------------	---

DSECU_POD_TYPEx

Syntax	<pre>#define DSECU_POD_TYPE0 #define DSECU_POD_TYPE1</pre>
---------------	--

Include file	<code>dsecucfg.h</code>
---------------------	-------------------------

Purpose	To specify the POD used to adapt the target.
----------------	--

Description	<p>This option specifies the kind of adapter which is used to access the target.</p> <p>Some address calculations must be adapted to the plug-on device (POD) that is used. Thus, the POD must be specified if the dSPACE Calibration and Bypassing Service is used for bypassing purposes. If bypassing and data acquisition are done in parallel, this option must be set for each tool individually.</p>
--------------------	---

Parameters

The following parameters are available as POD types for DSECU_POD_TYPE:

Predefined Symbol	Meaning
DSECU_DS1401_POD	POD for MicroAutoBox II
DSECU_DS4121_POD	POD for DS4121
DSECU_DS4120_POD	POD for DS4120
DSECU_RPCU_HOST_IF_POD	RapidPro system used as a stand-alone prototyping ECU

Example

```
#define DSECU_POD_TYPE0 DSECU_DS4121_POD
```

Related topics

Basics

[Basics on Configuration Options..... 88](#)

DSECU_PROCESSOR_TYPE

Syntax

```
#define DSECU_PROCESSOR_TYPE
```

Include file

dsecucfg.h

Purpose

To set the ECU byte order.

Description

Since the dSPACE Calibration and Bypassing Service can run on different kinds of ECUs, you must specify the byte order of the ECU's microcontroller.

You need this configuration option only if bypassing is enabled by DSECU_BYPASS.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
0	Little endian byte order
1	Big endian byte order

Example

```
#define DSECU_PROCESSOR_TYPE 1
```

Related topics**References**

DSECU_BYPASS..... 92

DSECU_BYTE_WRITE_SUPPORTED

Syntax

```
#define DSECU_BYTE_WRITE_SUPPORTED
```

Include file

dsecucfg.h

Purpose

To specify whether 8-bit write accesses to the tool RAM are supported by the calibration or bypass interface used.

Description

For example, there are PODs that do not support byte-wise ECU access, but only 16-bit ECU access. Using the DSECU_BYTE_WRITE_SUPPORTED option, 8-bit read or write ECU access is nevertheless enabled for these PODs. To read from the ECU, a bit mask specifying the needed bits is used. For write access, a macro is executed which performs a read/modify/write operation.

You need this configuration option only if bypassing is enabled by DSECU_BYPASS.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
0	8-bit write access to the tool RAM is not supported.
1	8-bit write access to the tool RAM is supported.

Example

```
#define DSECU_BYTE_WRITE_SUPPORTED 1
```

Related topics**References**

DSECU_BYPASS..... 92

Configuration of Calibration Features

Introduction

The dSPACE Calibration and Bypassing Service provides options to configure calibration features.

Where to go from here

Information in this section

DSECU_CAL_NUMBER_OF_PAGES.....	106
To specify the number of pages implemented in the ECU application.	
DSECU_CAL_PAGED_ADDRESSING.....	107
To enable or disable the paged address mode.	
DSECU_CAL_ABSOLUTE_ADDRESSING.....	108
To enable or disable the absolute address mode.	
DSECU_CAL_MEMORY_COPY_METHOD.....	108
To select the method used to copy memory.	

DSECU_CAL_NUMBER_OF_PAGES

Syntax

```
#define DSECU_CAL_NUMBER_OF_PAGES
```

Include file

dsecucfg.h

Purpose

To specify the number of calibration pages implemented in the ECU application.

Note

The page handling must be implemented according to your specific requirements.

Description

You need this configuration option only if calibration commands are enabled by `DSECU_CALIBRATION_COMMANDS`.

Parameters

<number of pages> Specifies the number of calibration pages on the ECU.

Example

```
#define DSECU_CAL_NUMBER_OF_PAGES 2
```

Related topics**References**

DSECU_CALIBRATION_COMMANDS..... 92

DSECU_CAL_PAGED_ADDRESSING

Syntax

```
#define DSECU_CAL_PAGED_ADDRESSING
```

Include file

dsecucfg.h

Purpose

To enable or disable the paged address mode for calibration.

Description

Address translation must be done by a customer-specific function which is included in the dSPACE Calibration and Bypassing Service Custom API. Both the paged and absolute address modes can be enabled in parallel. The tool automatically determines the address mode which must be used with a specific command.

You need this configuration option only if calibration commands are enabled by DSECU_CALIBRATION_COMMANDS.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables the paged address mode.
DSECU_DISABLED	Disables the paged address mode.

Example

```
#define DSECU_CAL_PAGED_ADDRESSING DSECU_DISABLED
```

Related topics**References**

DSECU_CALIBRATION_COMMANDS..... 92

DSECU_CAL_ABSOLUTE_ADDRESSING

Syntax

```
#define DSECU_CAL_ABSOLUTE_ADDRESSING
```

Include file

dsecucfg.h

Purpose

To enable or disable the absolute address mode for calibration.

Description

Address translation must be done by the tool. Since the tool does not know the location of the calibration pages in the ECU memory, this address mode is used only for memory locations that are not paged. Both the paged and absolute address modes can be enabled in parallel. The tool automatically determines the address mode which must be used with a specific command.

You need this configuration option only if calibration commands are enabled by DSECU_CALIBRATION_COMMANDS.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables the absolute address mode.
DSECU_DISABLED	Disables the absolute address mode.

Example

```
#define DSECU_CAL_ABSOLUTE_ADDRESSING DSECU_DISABLED
```

Related topics

References

DSECU_CALIBRATION_COMMANDS..... 92

DSECU_CAL_MEMORY_COPY_METHOD

Syntax

```
#define DSECU_CAL_MEMORY_COPY_METHOD
```

Include file

dsecucfg.h

Purpose To select the method used to copy memory.

Description In some scenarios, the dSPACE Calibration and Bypassing Service must be used to copy ECU memory from one location to another. This is used, for example, to configure the overlay RAM registers for calibration. This configuration option specifies the method used for copying the memory.

You need this configuration option only if calibration commands are enabled by `DSECU_CALIBRATION_COMMANDS`.

Parameters The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_MEMCPY_DEFAULT</code>	The library function <code>memcpy()</code> is used.
<code>DSECU_MEMCPY_CUSTOM</code>	The <code>dsecu_custom_copy()</code> function is used.
<code>DSECU_MEMCPY_REGISTER</code>	A special internal function optimized for register accesses is used. This method should be selected if the dSPACE Calibration and Bypassing Service is used for accessing overlay RAM calibration registers.

Example `#define DSECU_CAL_MEMORY_COPY_METHOD DSECU_MEMCPY_DEFAULT`

Related topics

References

<code>DSECU_CALIBRATION_COMMANDS</code>	92
<code>dsecu_custom_copy</code>	79

Configuration of DAQ and Bypassing Features

Introduction

You can use several configuration options to configure data acquisition (DAQ) and bypassing features.

Where to go from here

Information in this section

DSECU_SCS_LOCATION	111
To specify the location of the Service Configuration Section (SCS) copy.	
DSECU_SCSOFFSETADDRx	112
To specify the offset of the original Service Configuration Section (SCS).	
MORE_ATs_FOR_ONE_SRV	112
To enable or disable the use of several Address Tables per service call.	
DSECU_DOUBLE_BUFFER	113
To enable or disable the double buffer mechanism.	
DSECU_BLOCK_COPY	114
To enable or disable the block copy mechanism.	
DSECU_xxBIT_SUPPORT	115
To enable or disable the support of 64-bit, 32-bit, 16-bit and 8-bit data types.	
DSECU_BITS_SUPPORT	116
To enable or disable the support of bit modification.	
DSECU_ECU_WAITS	117
To enable or disable the wait mechanism.	
DSECU_FAILURE_CHECKING	118
To enable or disable the failure checking mechanism.	
DSECU_TIMEOUT_CUSTOM_FUNCTION	119
To enable or disable the custom time function for specifying the timeout.	
DSECU_TIMEOUT_TIMESTAMP_TYPE	121
To specify the data type of the time stamp used with the custom timeout mechanism.	
DSECU_TIMEOUT_SCALING_10US	122
To specify the base unit for the timeout in the wait mechanism.	
DSECU_FAILSAFE_PAGE	123
To enable or disable the fail-safe page.	
DSECU_CUSTOM_HW_TRIGGER	124
To enable or disable custom hardware trigger.	
DSECU_SUBINT_TYPE	125
To specify the type of subinterrupt handling.	

DSECU_SINT_NUMBER.....	126
To specify the number of subinterrupts.	
DSECU_MAX_SERVICE_COUNT.....	127
To specify the maximum number of service IDs.	

DSECU_SCS_LOCATION

Syntax

```
#define DSECU_SCS_LOCATION
```

Include file

dsecucfg.h

Purpose

To specify the location of the Service Configuration Section (SCS) copy.

Description

The dSPACE Calibration and Bypassing Service copies the SCS to another memory location. This can be either the normal RAM of the ECU, or the tool RAM.

If the SCS copy is located in the tool RAM, the amount of local RAM is minimized.

You need this configuration option only if data acquisition and bypassing are enabled by DSECU_DAQ_BYPASS_SERVICE.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
0	SCS copy is stored in the normal RAM.
1	SCS copy is stored in the dual-port RAM.

Example

```
#define DSECU_SCS_LOCATION 0
```

Related topics

References

DSECU_DAQ_BYPASS_SERVICE.....	91
-------------------------------	----

DSECU_SCSOFFSETADDRx

Syntax

```
#define DSECU_SCSOFFSETADDR0
#define DSECU_SCSOFFSETADDR1
```

Include file

dsecucfg.h

Purpose

To specify the offset of the original Service Configuration Section (SCS) in the tool RAM.

Description

The start address of the SCS is calculated by adding this offset to the start address of the service memory (specified by `DSECU_DPMEM_OFFSETADDRx`). Generally, the offset of the original SCS is set to 0.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECU_DAQ_BYPASS_SERVICE`.

Parameters

offset Specifies the offset of the original SCS within the service memory.

Example

```
#define DSECU_SCSOFFSETADDR0 0
```

Related topics

References

[DSECU_DAQ_BYPASS_SERVICE.....91](#)

MORE_ATs_FOR_ONE_SRV

Syntax

```
#define MORE_ATs_FOR_ONE_SRV
```

Include file

dsecucfg.h

Purpose

To enable or disable the use of several Address Tables per service call.

Description

This option enables or disables the use of more than one Address Tables per service ID. Each Address Table can be configured for one data direction. For example, if more than one Address Table per service ID is supported, an ECU service call first reads data and then writes data to the tool RAM, or vice versa. The tool can then assign more than one Address Table to one service ID. However, supporting more than one Address Table per service ID requires slightly more execution time, because the dSPACE Calibration and Bypassing Service must scan all the Address Tables. If only one Address Table is allowed, the service can skip the remaining Address Tables.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECU_DAQ_BYPASS_SERVICE`.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_ENABLED</code>	Enables more than one Address Table per service ID.
<code>DSECU_DISABLED</code>	Disables more than one Address Table per service ID.

Example

```
#define MORE_ATs_FOR_ONE_SRV DSECU_DISABLED
```

Related topics**References**

`DSECU_DAQ_BYPASS_SERVICE`..... 91

DSECU_DOUBLE_BUFFER

Syntax

```
#define DSECU_DOUBLE_BUFFER
```

Include file

`dsecucfg.h`

Purpose

To enable or disable the double buffer mechanism.

Description

The double buffer mechanism is used to ensure that only consistent data blocks are transferred to or from the tool RAM. If the mechanism is enabled, two buffers are used for data exchange between the ECU and the rapid control prototyping (RCP) system. If the option is set to `DSECU_DISABLED`, the double

buffer mechanism is not supported, and only one data buffer is used for data exchange.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECU_DAQ_BYPASS_SERVICE`.

For information on the double buffer mechanism, refer to [Extended Bypassing Mechanisms](#) on page 41.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_ENABLED</code>	Enables the double buffer mechanism.
<code>DSECU_DISABLED</code>	Disables the double buffer mechanism.

Example

```
#define DSECU_DOUBLE_BUFFER DSECU_DISABLED
```

Related topics**References**

[DSECU_DAQ_BYPASS_SERVICE.....91](#)

DSECU_BLOCK_COPY

Syntax

```
#define DSECU_BLOCK_COPY
```

Include file

`dsecucfg.h`

Purpose

To enable or disable data block transfers.

Description

This configuration option is a global setting for the dSPACE Calibration and Bypassing Service, that is, it is done for both bypassing and data acquisition.

Data block transfer is usually disabled for bypassing, because only single data values are transferred. This way, the size of the service code is minimized, and the code is faster. However, if the tool requires the dSPACE Calibration and Bypassing Service to transfer data blocks as well, you must enable the data block copy mechanism with `DSECU_BLOCK_COPY`.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECU_DAQ_BYPASS_SERVICE`.

For information on the block copy mechanism, refer to [Block Copy Mechanism](#) on page 58.

Note

If you perform bypassing via the RTI Bypass Blockset, data block transfers must be disabled.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_ENABLED</code>	Enables the block copy mechanism.
<code>DSECU_DISABLED</code>	Disables the block copy mechanism.

Example

```
#define DSECU_BLOCK_COPY DSECU_DISABLED
```

Related topics

References

[DSECU_DAQ_BYPASS_SERVICE..... 91](#)

DSECU_xxBIT_SUPPORT

Syntax

```
#define DSECU_64BIT_SUPPORT
#define DSECU_32BIT_SUPPORT
#define DSECU_16BIT_SUPPORT
#define DSECU_8BIT_SUPPORT
```

Include file

`dsecucfg.h`

Purpose

To enable or disable data transfer of 64-bit, 32-bit, 16-bit, or 8-bit.

Description

The options are enabled by default. If a data size is not used, for example, because it is not supported by the target processor, or because it is not used for data exchange between the ECU and the tool, you can disable the

corresponding option. This minimizes the compiled service code and makes it faster. If a data size remains enabled although it is not used, the only disadvantage is that the service execution time and the compiled code are increased. The `DSECU_64BIT_SUPPORT` option especially might require a floating-point library to be included, if the ECU processor does not feature a floating-point unit.

Enabled options are valid for all the tools connected to the ECU. Even if the transfer of a certain data type is used by only one tool, the corresponding option must be enabled, and the data type is available for every tool.

If 64-bit transfers are disabled, variables of this size can nevertheless be transferred by transferring two 32-bit values. If 8-bit data transfers are disabled, only an even number of bytes can be transferred.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECU_DAQ_BYPASS_SERVICE`.

Note

The RTI Bypass Blockset supports data transfer only up to 32-bit values.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_ENABLED</code>	Enables the transfer of 64-bit, 32-bit, 16-bit, or 8-bit values.
<code>DSECU_DISABLED</code>	Disables the transfer of 64-bit, 32-bit, 16-bit, or 8-bit values.

Example

```
#define DSECU_32BIT_SUPPORT DSECU_ENABLED
```

Related topics

References

[DSECU_DAQ_BYPASS_SERVICE..... 91](#)

DSECU_BITS_SUPPORT

Syntax

```
#define DSECU_BITS_SUPPORT
```

Include file

`dsecucfg.h`

Purpose	To enable or disable the writing of one or more bits to the ECU for bypassing.						
Description	<p>This option enables or disables the support for bit modification in the ECU. The transfer of bits from the ECU to the tool is not supported. For this direction, the whole byte containing the bit(s) must be read, and the other bits must be masked out on the host PC with a binary AND operation. Thus, enabling bit transfers makes sense only for bypassing.</p> <p>You need this configuration option only if bypassing is enabled by <code>DSECU_BYPASS</code>.</p>						
Parameters	<p>The following parameters are available to specify the option:</p> <table border="1"> <thead> <tr> <th>Predefined Symbol</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td><code>DSECU_ENABLED</code></td><td>Enables bit transfers.</td></tr> <tr> <td><code>DSECU_DISABLED</code></td><td>Disables bit transfers.</td></tr> </tbody> </table>	Predefined Symbol	Meaning	<code>DSECU_ENABLED</code>	Enables bit transfers.	<code>DSECU_DISABLED</code>	Disables bit transfers.
Predefined Symbol	Meaning						
<code>DSECU_ENABLED</code>	Enables bit transfers.						
<code>DSECU_DISABLED</code>	Disables bit transfers.						
Example	<pre>#define DSECU_BITS_SUPPORT DSECU_DISABLED</pre>						
Related topics	<p>References</p> <table border="1"> <tbody> <tr> <td><code>DSECU_BYPASS</code>.....</td><td>92</td></tr> </tbody> </table>	<code>DSECU_BYPASS</code>	92				
<code>DSECU_BYPASS</code>	92						

DSECU_ECU_WAITS

Syntax	<pre>#define DSECU_ECU_WAITS</pre>
Include file	<code>dsecucfg.h</code>
Purpose	To enable or disable the wait mechanism for bypassing.
Description	<p>The wait mechanism is used with bypassing to wait for a valid response from the tool (new data available), provided that the double buffer mechanism is enabled.</p> <p>You need this configuration option only if the double buffer mechanism is enabled by <code>DSECU_DOUBLE_BUFFER</code>.</p>

For information on the wait mechanism, refer to [Extended Bypassing Mechanisms](#) on page 41.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables the ECU wait mechanism.
DSECU_DISABLED	Disables the ECU wait mechanism.

Example

```
#define DSECU_ECU_WAITS DSECU_DISABLED
```

Related topics**References**

[DSECU_DOUBLE_BUFFER..... 113](#)

DSECU_FAILURE_CHECKING

Syntax

```
#define DSECU_FAILURE_CHECKING
```

Include file

dsecucfg.h

Purpose

To enable or disable the failure checking mechanism for bypassing.

Description

If enabled, the failure checking mechanism is supported for data communication from the tool to the ECU.

You need this configuration option only if the double buffer mechanism is enabled by `DSECU_DOUBLE_BUFFER`.

For information on the failure checking mechanism, refer to [Extended Bypassing Mechanisms](#) on page 41.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables the failure checking mechanism.
DSECU_DISABLED	Disables the failure checking mechanism.

Example

```
#define DSECUCHECKING DSECUDISABLED
```

Related topics**References**

DSECUDOUBLE_BUFFER..... 113

DSECUTIMEOUT_CUSTOM_FUNCTION

Syntax

```
#define DSECUTIMEOUT_CUSTOM_FUNCTION
```

Include file

dsecucfg.h

Purpose

To enable or disable the custom time stamp functionality for specifying the timeout.

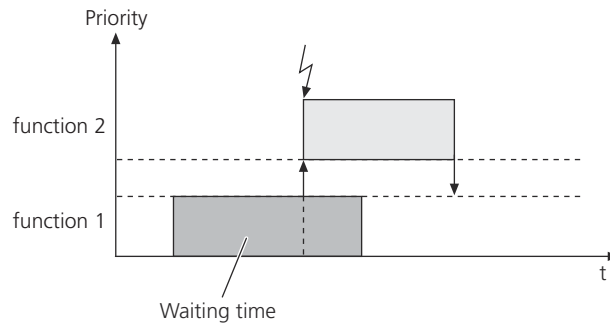
Description

This option lets you specify the mechanism used to detect a timeout in the wait mechanism.

- If this option is enabled, the timeout is calculated via the custom timeout mechanism. The DSECUTIMEOUT_TIMESTAMP_TYPE define is used to specify the timeout's data type.

With the custom timeout mechanism enabled, the waiting process of a function is not interrupted by the execution of another function, but continued simultaneously. That means interrupts do not extend the waiting time of a function. The absolute waiting period (i.e., the time from the start of the waiting process to its end) matches the timeout specified for the wait mechanism.

Example:

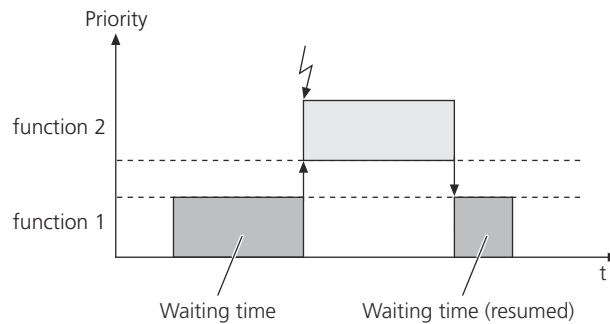


Function 1 starts its waiting process. When function 2 is triggered for execution, it starts execution. The waiting period of function 1 continues until a timeout occurs. If the timeout occurs while function 2 is being executed, function 1 will not continue waiting after function 2 has finished. If the timeout did not occur during function 2 execution, function 1 still waits after function 2 has finished.

- If this option is disabled, timeout detection is without custom time stamp functionality. You must configure a timeout scaling factor in the service configuration, using the `DSECU_TIMEOUT_SCALING_10US` define.

Without the custom timeout mechanism, the waiting process of a function is interrupted by the execution of a function with higher priority. The absolute waiting period is extended by the execution time of the function with higher priority.

Example:



Function 1 starts its waiting process. When function 2 is ready for execution, the waiting process of function 1 is interrupted until function 2 has finished. Function 1 then resumes its waiting process.

If the `DSECU_TIMEOUT_CUSTOM_FUNCTION` define is not set at all, the custom timeout functionality is disabled.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECU_ENABLED</code>	Enables the custom timeout functionality.
<code>DSECU_DISABLED</code>	Disables the custom timeout functionality.

Example

```
#define DSECU_TIMEOUT_CUSTOM_FUNCTION DSECU_DISABLED
```

Related topics**References**

DSECU_TIMEOUT_SCALING_10US.....	122
DSECU_TIMEOUT_TIMESTAMP_TYPE.....	121

DSECU_TIMEOUT_TIMESTAMP_TYPE

Syntax

```
#define DSECU_TIMEOUT_TIMESTAMP_TYPE
```

Include file

dsecucfg.h

Purpose

To specify the data type of the time stamp used with the custom timeout mechanism.

Description

The time stamp can have any data type.

You need this configuration option only if the following configuration options are enabled:

- The custom timeout mechanism must be enabled by DSECU_TIMEOUT_CUSTOM_FUNCTION.
- The wait mechanism must be enabled by DSECU_ECU_WAITS.
- The double buffer mechanism must be enabled by DSECU_DOUBLE_BUFFER.
- Data acquisition and bypassing must be enabled by DSECU_DAQ_BYPASS_SERVICE.
- Bypassing must be enabled by DSECU_BYPASS.

Parameters

... Data type of the time stamp.

Example

```
#define DSECU_TIMEOUT_TIMESTAMP_TYPE DSECU_UInt32
```

Related topics

References

dsecu_custom_timeout_pending_check.....	86
DSECU_TIMEOUT_CUSTOM_FUNCTION.....	119

DSECU_TIMEOUT_SCALING_10US

Syntax

```
#define DSECU_TIMEOUT_SCALING_10US
```

Include file

dsecucfg.h

Purpose

To specify the base unit for the timeout in the wait mechanism.

Description

You can specify a scaling factor for a timeout with the double buffer mechanism. The timeout can only be used if the double buffer mechanism and the ECU wait mechanism are enabled.

A timeout is implemented as a **while** loop in the dSPACE Calibration and Bypassing Service. The timeout expires when the maximum number of loops is reached. The maximum number of loops is determined by the timeout value specified in the Address Table Header multiplied by the scaling factor specified in this option. The number of loops processed within a certain time period depends on the ECU, so the specific scaling factor must be determined by testing. The timeout value is entered in the Address Table Header during table generation by the AutoBox/MicroAutoBox II.

You need this configuration option only if the **DSECU_TIMEOUT_CUSTOM_FUNCTION** option is disabled and if the following configuration options are enabled:

- The wait mechanism must be enabled by **DSECU_ECU_WAITS**.
- The double buffer mechanism must be enabled by **DSECU_DOUBLE_BUFFER**.
- Data acquisition and bypassing must be enabled by **DSECU_DAQ_BYPASS_SERVICE**.
- Bypassing must be enabled by **DSECU_BYPASS**.

Note

The specified timeout matches the real timeout only if this define is set properly.

Parameters	<scaling factor> Factor for scaling the timeout.										
Example	<pre>#define DSECU_TIMEOUT_SCALING_10US 1000</pre>										
Related topics	References <table> <tr> <td>DSECU_BYPASS.....</td><td>92</td></tr> <tr> <td>DSECU_DAQ_BYPASS_SERVICE.....</td><td>91</td></tr> <tr> <td>DSECU_DOUBLE_BUFFER.....</td><td>113</td></tr> <tr> <td>DSECU_ECU_WAITS.....</td><td>117</td></tr> <tr> <td>DSECU_TIMEOUT_CUSTOM_FUNCTION.....</td><td>119</td></tr> </table>	DSECU_BYPASS.....	92	DSECU_DAQ_BYPASS_SERVICE.....	91	DSECU_DOUBLE_BUFFER.....	113	DSECU_ECU_WAITS.....	117	DSECU_TIMEOUT_CUSTOM_FUNCTION.....	119
DSECU_BYPASS.....	92										
DSECU_DAQ_BYPASS_SERVICE.....	91										
DSECU_DOUBLE_BUFFER.....	113										
DSECU_ECU_WAITS.....	117										
DSECU_TIMEOUT_CUSTOM_FUNCTION.....	119										

DSECU_FAILSAFE_PAGE

Syntax	<pre>#define DSECU_FAILSAFE_PAGE</pre>
Include file	dsecucfg.h
Purpose	To enable or disable the fail-safe page.
Description	<p>The fail-safe mechanism is used to provide valid data (fail-safe data) when the tool has failed to provide new data to the ECU more often than indicated in the failure limit value. The failure checking mechanism and the double buffer mechanism must be enabled to make this mechanism available.</p> <p>When the fail-safe mechanism is enabled, the fail-safe flag and the failure limit have to be set in the Address Table Header. If the maximum number of failures is reached, the fail-safe data is read. If the fail-safe mechanism is disabled, the service call will return an error.</p> <p>You need this configuration option only if the following configuration options are enabled:</p> <ul style="list-style-type: none"> ▪ The failure checking mechanism must be enabled by DSECU_FAILURE_CHECKING. ▪ The double buffer mechanism must be enabled by DSECU_DOUBLE_BUFFER. ▪ Data acquisition and bypassing must be enabled by DSECU_DAQ_BYPASS_SERVICE. ▪ Bypassing must be enabled by DSECU_BYPASS.

Note

Currently, the fail-safe mechanism is not supported by the RTI Bypass Blockset.

For information on the fail-safe mechanism, refer to [Extended Bypassing Mechanisms](#) on page 41.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_ENABLED	Enables the fail-safe page.
DSECU_DISABLED	Disables the fail-safe page.

Example

```
#define DSECU_FAILSAFE_PAGE DSECU_DISABLED
```

Related topics**References**

DSECU_BYPASS.....	92
DSECU_DAQ_BYPASS_SERVICE.....	91
DSECU_DOUBLE_BUFFER.....	113
DSECU_FAILURE_CHECKING.....	118

DSECU_CUSTOM_HW_TRIGGER

Syntax

```
#define DSECU_CUSTOM_HW_TRIGGER
```

Include file

dsecucfg.h

Purpose

To enable or disable custom hardware triggers.

Description

By default, a hardware interrupt is triggered on the tool by writing to an address in the dual-port memory. This is the case for bypassing PODs (plug-on devices). If no dual-port memory exists, another interrupt trigger method must be used. This can be a digital I/O pin of the ECU processor, for example.

If this option is enabled, the function `dsecu_custom_trigger_hw_int()` is called every time a hardware interrupt is to be triggered.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECUCFG_BYPASS_SERVICE`.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
<code>DSECUCFG_ENABLED</code>	Enables custom hardware triggers.
<code>DSECUCFG_DISABLED</code>	Disables custom hardware triggers.

Example

```
#define DSECUCFG_CUSTOM_HW_TRIGGER DSECUCFG_DISABLED
```

Related topics

References

dsecucfg_custom_trigger_hw_int	84
DSECUCFG_BYPASS_SERVICE	91

DSECUCFG_SUBINT_TYPE

Syntax

```
#define DSECUCFG_SUBINT_TYPE
```

Include file

`dsecucfg.h`

Purpose

To specify the method for subinterrupt handling.

Description

In most cases, there is only one hardware interrupt line from the ECU to the tool. To handle multiple interrupts, subinterrupt handling is needed. The methods differ in the number of bits that are required for handling a subinterrupt:

- The word-based subinterrupt handling uses 16 bits per subinterrupt. This method has always been used so far, so it offers high compatibility.
- The byte-based subinterrupt handling uses eight bits per subinterrupt. This method is a cross between the other two methods.
- With bit-based subinterrupt handling, one bit is used per subinterrupt.

You need this configuration option only if data acquisition and bypassing are enabled by `DSECUCFG_BYPASS_SERVICE`.

For further information on the subinterrupt handling mechanism, refer to [Subinterrupt Handling Mechanism](#) on page 55.

Parameters

The following parameters are available to specify the option:

Predefined Symbol	Meaning
DSECU_SUBINT_1_BIT	Specifies that 1 bit is used to handle a subinterrupt.
DSECU_SUBINT_8_BIT	Specifies that 8 bits are used to handle a subinterrupt.
DSECU_SUBINT_16_BIT	Specifies that 16 bits are used to handle a subinterrupt.

Example

```
#define DSECU_SUBINT_TYPE DSECU_SUBINT_16_BIT
```

Related topics**Basics**

Subinterrupt Handling Mechanism..... 55

References

DSECU_DAQ_BYPASS_SERVICE..... 91
DSECU_SINT_NUMBER..... 126

DSECU_SINT_NUMBER

Syntax

```
#define DSECU_SINT_NUMBER
```

Include file

dsecucfg.h

Purpose

To specify the number of subinterrupts.

Description

You can adapt the number of subinterrupts to the ECU application. The more subinterrupts are used, the more memory is required to handle them, and the longer it takes to read out the memory.

Parameters

<number of subinterrupts> Specifies the number of subinterrupts.

Example

```
#define DSECU_SINT_NUMBER 16
```

Related topics	Basics
	Subinterrupt Handling Mechanism..... 55
	References
	DSECU_SUBINT_TYPE..... 125

DSECU_MAX_SERVICE_COUNT

Syntax	<code>#define DSECU_MAX_SERVICE_COUNT</code>
Include file	dsecucfg.h
Purpose	To specify the maximum number of service IDs.
Description	This define should be adjusted if more than 255 service IDs are needed. The more service IDs are supported, the more tool RAM is required to handle them.
Parameters	<number of service IDs> Specifies the maximum number of service IDs.
Example	<code>#define DSECU_MAX_SERVICE_COUNT 1023</code>
Related topics	Basics
	Basics on the Data Structures..... 34

Configuration of ECU Flash Programming Features

Introduction

The dSPACE Calibration and Bypassing Service provides several options for configuring ECU flash programming features.

Where to go from here

Information in this section

Overview of Options for Configuring ECU Flash Programming Features.....	129
Depending on what you want to do, different options are required.	
DSECU_MAILBOX_ADDRx.....	129
To specify the start address of the dSPACE mailbox which is required for communication between the ECU and the dSPACE ECU Flash Programming Tool.	
DSECU_FLASH_MAX_SECTORS.....	130
To specify the maximum number of sectors that can be flashed.	
DSECU_FLASH_SECTOR_LIST.....	131
To specify a list with configuration data for each flash sector.	
DSECU_FLASH_MAX_CODECHECKS.....	132
To specify the maximum number of code checks.	
DSECU_FLASH_CODECHECK_LIST.....	133
To specify a list of code check data.	
DSECU_APPLICATION_LOCK_INIT.....	134
To initialize ECU application locking during flash kernel download.	
DSECU_APPLICATION_LOCK_LOOP.....	135
To lock the ECU application during the download of the flash kernel.	
DSECU_APPLICATION_VALID_CHECK.....	135
To enable or disable the boot loader check for a valid ECU application.	
DSECU_APPLICATION_VALID_ID_ADDRESS.....	136
To specify the location of the flash valid identifier in the ECU flash memory.	
DSECU_APPLICATION_VALID_ID_VALUE.....	137
To specify a fixed value indicating that the last ECU flash programming operation was completed successfully.	

Overview of Options for Configuring ECU Flash Programming Features

Introduction

Depending on what you want to do, different options are required.

Option overview

Options for building a custom flash kernel (Necessary only if no flash kernel provided by dSPACE is to be used) To build a custom flash kernel, the following options are required:

- DSECU_MAILBOX_ADDRx
- DSECU_FLASH_MAX_SECTORS
- DSECU_FLASH_SECTOR_LIST
- DSECU_FLASH_MAX_CODECHECKS
- DSECU_FLASH_CODECHECK_LIST

Note

The code check configuration settings are made in the dSPACE ECU Flash Programming Tool. Thus, the DSECU_FLASH_MAX_CODECHECKS and DSECU_FLASH_CODECHECK_LIST configuration options are obsolete.

Options for activating ECU flash programming on the ECU To perform ECU flash programming, the following configuration options of the dsECUbootcfg.h file are required (regardless of whether a flash kernel provided by dSPACE or a self-programmed flash kernel is used):

- DSECU_MAILBOX_ADDRx
- DSECU_APPLICATION_LOCK_INIT
- DSECU_APPLICATION_LOCK_LOOP
- DSECU_APPLICATION_VALID_CHECK
- DSECU_APPLICATION_VALID_ID_ADDRESS
- DSECU_APPLICATION_VALID_ID_VALUE

Related topics

Basics

[ECU Flash Programming Mechanism..... 59](#)

DSECU_MAILBOX_ADDRx

Syntax

```
#define DSECU_MAILBOX_ADDR0
#define DSECU_MAILBOX_ADDR1
```

Include file	dsECUbootcfg.h
Purpose	To specify the start address of the dSPACE mailbox. The mailbox is used for communication between the ECU and the dSPACE ECU Flash Programming Tool.
Description	<p>Via the dSPACE mailbox, the dSPACE ECU Flash Programming Tool indicates a pending ECU flashing request to the ECU. During ECU flash programming, the mailbox is used to transfer the erase and program commands from the flash tool to the ECU.</p> <div> <p>Note</p> <p>An DSECU_MAILBOX_ADDR entry is always 32-bit aligned. This memory area must be used for ECU flash programming only. Since the flash tool can write to the mailbox at any time, you must ensure that the memory area specified for the dSPACE mailbox is not used by the ECU application or the ECU boot code.</p> </div>
Parameters	<p><start address of mailbox> Specifies the start address of the dSPACE mailbox.</p>
Example	<pre>#define DSECU_MAILBOX_ADDR0 0xFFFF0000</pre>
Related topics	<p>Basics</p> <p>Overview of Options for Configuring ECU Flash Programming Features..... 129</p>

DSECU_FLASH_MAX_SECTORS

Syntax	<pre>#define DSECU_FLASH_MAX_SECTORS</pre>
Include file	dsecucfg.h
Purpose	To specify the maximum number of sectors that can be flashed to the ECU.

Description	<p>The size must match the length of <code>DSECU_FLASH_SECTOR_LIST</code>.</p> <p>You need this configuration option only if the flashing commands are enabled by <code>DSECU_FLASH_COMMANDS</code>.</p>
Parameters	<p><number of sectors> Specifies the number of sectors that can be flashed.</p>
Example	<pre>#define DSECU_FLASH_MAX_SECTORS 2</pre>
Related topics	<p>Basics</p> <p>Overview of Options for Configuring ECU Flash Programming Features..... 129</p> <p>References</p> <p>DSECU_FLASH_COMMANDS..... 94</p>

DSECU_FLASH_SECTOR_LIST

Syntax	<pre>#define DSECU_FLASH_SECTOR_LIST</pre>
Include file	<p>dsecucfg.h</p>
Purpose	<p>To specify the list with configuration data for every flash sector.</p>

Description	<p>This option specifies the configuration for a list of flash sectors on the ECU. The list must contain exactly <code>DSECU_FLASH_MAX_SECTORS</code> entries.</p> <p>Each sector entry consists of the following elements:</p> <ul style="list-style-type: none"> ▪ Address of the sector <ul style="list-style-type: none"> This is the address of the sector within the ECU's address space. ▪ Length of the sector (in bytes) ▪ Sector properties <ul style="list-style-type: none"> Currently, <code>DSECU_FLASH_ABSOLUTE_ADDRESS</code> must be used. ▪ Flash program granularity (in bytes) <ul style="list-style-type: none"> This value specifies the minimum size of an element that can be flashed at a time.
--------------------	--

- Erase sequence number
Sectors are erased in ascending order of the erase sequence numbers.
- Program sequence number
Sectors are programmed in ascending order of the program sequence numbers.
- Memory type
The memory type is displayed in the flash tool for information. The following memory types are available:
 - DSECU_MEMTYPE_EXT_FLASH
 - DSECU_MEMTYPE_EXT_RAM
 - DSECU_MEMTYPE_EXT_NVRAM
 - DSECU_MEMTYPE_INT_FLASH
 - DSECU_MEMTYPE_INT_RAM
 - DSECU_MEMTYPE_INT_NVRAM

You need this configuration option only if the flashing commands are enabled by DSECU_FLASH_COMMANDS.

Parameters ... Specifies the list of flash sectors in the ECU.

Example

```
#define DSECU_FLASH_SECTOR_LIST
{
    {0x100000,0x10000,DSECU_FLASH_ABSOLUTE_ADDRESS,4,0,1,
      DSECU_MEMTYPE_INT_FLASH}, \
    {0x400000,0x20000,DSECU_FLASH_ABSOLUTE_ADDRESS,128,2,3,
      DSECU_MEMTYPE_INT_FLASH} \
}
```

Related topics

Basics

[Overview of Options for Configuring ECU Flash Programming Features..... 129](#)

References

[DSECU_FLASH_COMMANDS..... 94](#)

DSECU_FLASH_MAX_CODECHECKS

Syntax

```
#define DSECU_FLASH_MAX_CODECHECKS
```

Include file	dsecucfg.h
Purpose	To specify the maximum number of code checks.
Description	<p>This option specifies the number of code checks defined in this configuration template.</p> <div> <p>Note</p> <p>This option is obsolete. The code check configuration settings are now made in the dSPACE ECU Flash Programming Tool. You should therefore set the value of this define to 0.</p> </div>
Parameters	<p><number of code checks> Specifies the number of code check data packages.</p>
Example	<pre>#define DSECU_FLASH_MAX_CODECHECKS 0</pre>
Related topics	<p>Basics</p> <div> <p>Overview of Options for Configuring ECU Flash Programming Features..... 129</p> </div>

DSECU_FLASH_CODECHECK_LIST

Syntax	<pre>#define DSECU_FLASH_CODECHECK_LIST</pre>
Include file	dsecucfg.h
Purpose	To specify the configuration for a list of data packages used for the code check.

Description This option specifies the configuration for a list of code check data packages.

Note

This option is obsolete. The code check configuration settings are now made in the dSPACE ECU Flash Programming Tool. You should therefore set the value of this define to 0.

Parameters ... Specifies the list of code check data packages.

Example

```
#define DSECU_FLASH_CODECHECK_LIST {}
```

Related topics**Basics**

[Overview of Options for Configuring ECU Flash Programming Features..... 129](#)

DSECU_APPLICATION_LOCK_INIT

Syntax

```
#define DSECU_APPLICATION_LOCK_INIT
```

Include file dsECUbootcfg.h

Purpose To prepare the ECU application for downloading the flash kernel.

Description DSECU_APPLICATION_LOCK_INIT is called once, when the `dsecuboot_check` function has recognized an ECU flashing request. The macro is executed just before the flash kernel is downloaded to the ECU's RAM. It is used to prepare the ECU application for downloading the flash kernel.

DSECU_APPLICATION_LOCK_INIT can be used, for example, to disable interrupts.

Parameters ... Code that is executed just before the flash kernel is downloaded.

Example

```
#define DSECU_APPLICATION_LOCK_INIT() { DISABLE_INTERRUPTS();}
```

Related topics**Basics**

Overview of Options for Configuring ECU Flash Programming Features..... 129

DSECU_APPLICATION_LOCK_LOOP

Syntax

```
#define DSECU_APPLICATION_LOCK_LOOP
```

Include file

dsECUbootcfg.h

Purpose

To execute custom commands during the download of the flash kernel.

Description

During the download of the flash kernel, the ECU application is locked, which means it remains in an idle loop. This is necessary because the flash kernel is written to RAM memory at the risk of overwriting other important data (for example, program variables). The macro is called repeatedly by the `dsecuboot_check` function until the flash kernel has been completely transferred to the ECU memory.

DSECU_APPLICATION_LOCK_LOOP may be used, for example, to service the watchdog.

Parameters

... Code that is executed during the lock loop.

Example

```
#define DSECU_APPLICATION_LOCK_LOOP() { SERVICE_WATCHDOG(); }
```

Related topics**Basics**

Overview of Options for Configuring ECU Flash Programming Features..... 129

DSECU_APPLICATION_VALID_CHECK

Syntax

```
#define DSECU_APPLICATION_VALID_CHECK
```

Include file	dsECUbootcfg.h						
Purpose	To enable or disable the boot loader check to find out if the flash memory contains a valid ECU application.						
Description	<p>The boot loader check for a valid ECU application is used to check whether the last ECU flash programming operation was completed successfully or not. When the check is enabled, the boot loader checks for the "flash valid identifier" in the ECU flash memory (at the address specified by <code>DSECUC_APPLICATION_VALID_ID_ADDRESS</code>). The value in the ECU flash memory is compared to the value specified by <code>DSECUC_APPLICATION_VALID_ID_VALUE</code>. If the values differ, the last ECU flash programming operation is assumed to be invalid or incomplete, so the ECU application is not started and the ECU stays in flash-programming mode. The ECU application will not start until another ECU flash programming operation is completed successfully.</p>						
Parameters	<p>The following parameters are available to specify the option:</p> <table border="1"> <thead> <tr> <th>Predefined Symbol</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td><code>DSECUC_ENABLED</code></td><td>Enables the boot loader check for a valid ECU application.</td></tr> <tr> <td><code>DSECUC_DISABLED</code></td><td>Disables the boot loader check for a valid ECU application.</td></tr> </tbody> </table>	Predefined Symbol	Meaning	<code>DSECUC_ENABLED</code>	Enables the boot loader check for a valid ECU application.	<code>DSECUC_DISABLED</code>	Disables the boot loader check for a valid ECU application.
Predefined Symbol	Meaning						
<code>DSECUC_ENABLED</code>	Enables the boot loader check for a valid ECU application.						
<code>DSECUC_DISABLED</code>	Disables the boot loader check for a valid ECU application.						
Example	<pre>#define DSECUC_APPLICATION_VALID_CHECK DSECUC_ENABLED</pre>						
Related topics	<p>Basics</p> <p>Overview of Options for Configuring ECU Flash Programming Features..... 129</p>						

DSECUC_APPLICATION_VALID_ID_ADDRESS

Syntax	<pre>#define DSECUC_APPLICATION_VALID_ID_ADDRESS</pre>
Include file	dsECUbootcfg.h

Purpose	To specify the location of the "flash valid identifier" in the ECU flash memory.
Description	<p>This define specifies the address of the flash valid identifier in the ECU flash memory. The identifier is always 4 bytes long.</p> <p>You can specify any location within a memory segment of CODE segment type in the ECU application, or the address of a specific ID you introduced for this purpose. The memory segment containing the identifier will be reprogrammed with every flash operation.</p> <p>You need this configuration option only if the boot loader check for a valid ECU application is enabled by <code>DSECUC_APPLICATION_VALID_CHECK</code>.</p>
Parameters	address Specifies the address of the flash valid identifier.
Example	<pre>#define DSECUC_APPLICATION_VALID_ID_ADDRESS(0x4100)</pre>
Related topics	<p>Basics</p> <p>Overview of Options for Configuring ECU Flash Programming Features..... 129</p> <p>References</p> <p>DSECUC_APPLICATION_VALID_CHECK..... 135</p>

DSECUC_APPLICATION_VALID_ID_VALUE

Syntax	<pre>#define DSECUC_APPLICATION_VALID_ID_VALUE</pre>
Include file	<code>dsECUbootcfg.h</code>
Purpose	To specify a fixed 32-bit value indicating that the last ECU flash programming operation was completed successfully.
Description	With the boot loader check for a valid ECU application enabled, the value found at <code>DSECUC_APPLICATION_VALID_ID_ADDRESS</code> in the ECU flash memory is compared to the value specified in this configuration option. If the values are equal, the last flash programming operation was completed successfully. If they

differ, the ECU flash programming operation is assumed to be invalid or incomplete.

Note

- If you specify a fixed value, make sure that your ECU application contains exactly that value at the "flash valid identifier" location (specified by `DSECUC_APPLICATION_VALID_ID_ADDRESS`).
- If you do not specify a fixed value with this configuration option, any value found in the ECU flash memory at the address specified by `DSECUC_APPLICATION_VALID_ID_ADDRESS` is accepted as a valid value, except for the values `0x0` and `0xFFFFFFFF`.

You need this configuration option only if the boot loader check for a valid ECU application is enabled by `DSECUC_APPLICATION_VALID_CHECK`.

Parameters

<value> Specifies the fixed value indicating a successful ECU flash programming operation.

Example

```
#define DSECUC_APPLICATION_VALID_ID_VALUE(0x12345678)
```

Related topics**Basics**

[Overview of Options for Configuring ECU Flash Programming Features..... 129](#)

References

[DSECUC_APPLICATION_VALID_CHECK..... 135](#)

dSPACE Calibration and Bypassing Service Interface Description

Introduction

To communicate with an ECU or RapidPro system used as a stand-alone prototyping ECU with integrated dSPACE Calibration and Bypassing Service, the interface used for calibration, measurement or bypassing must be described.

Interface Description

Introduction

ASAM MCD-2 MC (A2L) files contain a variable description of an ECU or RapidPro system used as a stand-alone prototyping ECU. To access an ECU or RapidPro system via ControlDesk for calibration and measurement, or to bypass an ECU via DPMEM using the RTI Bypass Blockset, the interface provided by the dSPACE Calibration and Bypassing Service must be described.

Required interface-specific information

The A2L file must include an **IF_DATA** element specific to the calibration, measurement or bypass interface used. The **IF_DATA** element contains interface-specific information. This information must comply with a special format, which is described in ASAP2 Meta Language (AML) files. The AML specification of the interface must also be contained in the A2L file.

The **IF_DATA** elements describing the dSPACE Calibration and Bypassing Service implementation must be set to the configuration defined in **dsECUcfg.h**. For information on how to configure the dSPACE Calibration and Bypassing Service, refer to [dSPACE Calibration and Bypassing Service Configuration Options](#) on page 87.

Note

Differences between the service configuration and the interface description may prevent the calibration/measurement or bypassing tool from accessing the ECU.

Note

To access one ECU with the DCI-GSI2 using two tools in parallel, for example, to perform measurement/calibration and bypassing at the same time, it is sufficient to configure one instance of the dSPACE Calibration and Bypassing Service. In other cases, two instances of the dSPACE Calibration and Bypassing Service must be configured to access one ECU using two tools in parallel. Then each tool-specific **IF_DATA** element must refer to a different instance of the dSPACE Calibration and Bypassing Service. In particular, the Tool RAM configurations must be different for the tools to ensure that each tool accesses a different tool RAM. For information on how to configure the dSPACE Calibration and Bypassing Service to be accessible from two tools at the same time, refer to [dSPACE Calibration and Bypassing Service Configuration Options](#) on page 87.

For information on the interface-specific **IF_DATA** elements and details on their data format, refer to the [Interface Description Data Reference](#) .

Related topics**Basics**

[Basics on Configuration Options](#)..... 88

A

Address Table 38
AT 38

B

bypassing
 buffer synchronization mechanism 43
 double buffer mechanism 42
 fail-safe mechanism 43
 failure checking mechanism 43
 wait mechanism 42

C

Common Program Data folder 8

D

data types for dSPACE Calibration and Bypassing Service 24
Documents folder 8
DSECU_APPLICATION_LOCK_INIT 134
DSECU_APPLICATION_LOCK_LOOP 135
DSECU_APPLICATION_VALID_CHECK 135
DSECU_APPLICATION_VALID_ID_ADDRESS 136
DSECU_APPLICATION_VALID_ID_VALUE 137
DSECU_BITS_SUPPORT 116
DSECU_BLOCK_COPY 114
DSECU_BYPASS 92
DSECU_BYTE_WRITE_SUPPORTED 105
DSECU_CAL_ABSOLUTE_ADDRESSING 108
DSECU_CAL_MEMORY_COPY_METHOD 108
DSECU_CAL_NUMBER_OF_PAGES 106
DSECU_CAL_PAGED_ADDRESSING 107
DSECU_CALIBRATION_COMMANDS 92
dsecu_custom_cal_absolute_access 81
dsecu_custom_cal_page_switch 80
dsecu_custom_cal_paged_access 82
dsecu_custom_cal_paged_access_finished 83
dsecu_custom_command 78
DSECU_CUSTOM_COMMANDS 93
dsecu_custom_copy 79
dsecu_custom_flash_decode 76
dsecu_custom_flash_erase_range 74
dsecu_custom_flash_finish 77
dsecu_custom_flash_program_range 75
dsecu_custom_flash_program_start 73
DSECU_CUSTOM_HW_TRIGGER 124
dsecu_custom_timeout_pending_check 86
dsecu_custom_timeout_timestamp_get 85
dsecu_custom_trigger_hw_int 84
DSECU_DAQ_BYPASS_SERVICE 91
DSECU_DOUBLE_BUFFER 113
DSECU_DPMEM_OFFSETADDRx 101
DSECU_DPMEM_SIZEx 102
DSECU_ECU_WAITS 117
DSECU_FAILSAFE_PAGE 123
DSECU_FAILURE_CHECKING 118
DSECU_FLASH_CODECHECK_LIST 133
DSECU_FLASH_COMMANDS 94

DSECU_FLASH_MAX_CODECHECKS 132
DSECU_FLASH_MAX_SECTORS 130
DSECU_FLASH_SECTOR_LIST 131
DSECU_INLINE 101
DSECU_INT_RESTORE 99
DSECU_INT_SAVE_AND_DISABLE 98
DSECU_INT_STATUS_VAR_DECL 97
DSECU_MAILBOX_ADDRx 129
DSECU_MAX_SERVICE_COUNT 127
DSECU_NUMBER_TOOLS 96
DSECU_POD_TYPEx 103
DSECU_POINTER_QUALIFIER 100
DSECU_PROCESSOR_TYPE 104
DSECU_SCS_LOCATION 111
DSECU_SCSOFFSETADDRx 112
dsecu_service 64
dsecu_service_background 67
dsecu_service_buffer_sync 68
dsecu_service_init 64
DSECU_SERVICE_REMOVED 90
dsecu_service_state 66
DSECU_SINT_NUMBER 126
DSECU_SUBINT_TYPE 125
DSECU_TIMEOUT_CUSTOM_FUNCTION 119
DSECU_TIMEOUT_SCALING_10US 122
DSECU_TIMEOUT_TIMESTAMP_TYPE 121
DSECU_xxBIT_SUPPORT 115
dsecuboot_check 69
dSPACE Calibration and Bypassing Service
 alive and version information mechanism 47
 API 25
 background service 26
 block copy mechanism 58
 boot check function 26
 configuration 87
 configuration of calibration features 106
 configuration of DAQ and bypassing features 110
 configuration of ECU flash programming features 128
 configuration of enabling/disabling features 90
 configuration of general features 96
 Custom API 71
 data structures 34
 data types 24
 dSPACE ECU Flash Programming Tool 15
 ECU flash programming mechanism 59
 file structure 22
 foreground service 26
 initialization 26
 integration in ECU code 27
 license agreement 9
 requirements
 bypassing 15
 calibration and data acquisition 14
 ECU flash programming 15
 software layers 20
 starting the flash kernel 26
 subinterrupt handling mechanism 55
 system components 15

dSPACE ECU Flash Programming Tool 15

E

ETPS 39
Extended Table Pointer Section 39

F

flash kernel 26

I

integrating the dSPACE Calibration and Bypassing Service 25

L

license agreement 9
license agreement for dSPACE Calibration and Bypassing Service 9
Local Program Data folder 8

M

MORE_ATs_FOR_ONE_SRV 112

S

SCS 37
service configuration
 dSPACE Calibration and Bypassing Service 87
Service Configuration Section 37
Service Pointer Table 39
SPT 39

T

Table Pointer Section 37
TPS 37

