

Generic Serial Interface (DCI-GSI2)

DCI-GSI2 Setup Application Note

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.

© 2011 - 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 Application Note	7
Safety Precautions	11
Warning About Using a DCI-GSI2.....	11
Preparing the DCI-GSI2	15
Initial Setup of the DCI-GSI2.....	16
Workflow for Initial Setup of the DCI-GSI2.....	16
Configuring the DCI-GSI2.....	18
Basics on Configuring a DCI-GSI2 Using the DCI Configuration Tool.....	18
Configuring and Optimizing the DCI-GSI2 Settings.....	19
Preparing the A2L File for the DCI-GSI2	21
Basics on Preparing the A2L File for the DCI-GSI2.....	21
Setting Up a Bypass With a DCI-GSI2	23
Introduction to Setting Up a Bypass with a DCI-GSI2.....	24
When to Use the DCI-GSI2.....	24
Bypassing With or Without an ECU Service.....	25
Workflow for Setting Up a Service-Based Bypass.....	26
Integrating the ECU Service.....	28
Manual Integration of the Source Code.....	28
Basics on Manual Source Code Integration.....	29
Obtaining the Service Source Code.....	30
Integrating the Sources Into the ECU Software Project.....	31
Configuring the Service Parameters.....	33
Customizing the Service.....	38
Integrating the Service Functions.....	39
Service Integration With the ECU Interface Manager.....	41
Basics on Service Integration via ECU Interface Manager.....	41
Integrating the Service Into the ECU Binary Code.....	43

Configuring the DCI-GSI2.....	47
Basics on DCI-GSI2 Configuration.....	47
Manual Service Parameter Configuration.....	48
Service Parameter Configuration via A2L File.....	51
Testing the Integrated Service.....	53
Testing the Service Functionality.....	53
A2L File Adaptation.....	55
Adapting or Generating an A2L File.....	55
Bypass Model Configuration.....	57
Configuring the Bypass Model.....	57
 Target Processor-Specific Configuration Settings	 59
Cypress MB9D560 Family.....	60
Basics on the Cypress MB9D560 Family.....	60
Generic Configuration Settings for the Cypress MB9D560 Target Processor Family.....	60
Freescall/NXP MPC55xx/MPC56xx Family and STMicroelectronics SPC55xx/SPC56xx Family.....	62
Basics on the Freescall/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families.....	62
Generic Configuration Settings for the Freescall/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families.....	63
Processor-Specific Configuration Settings (Freescall/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Families).....	64
Freescall/NXP MPC57xx Family and STMicroelectronics SPC57xx/SPC58xx Family.....	67
Basics on the Freescall/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families.....	67
Generic Configuration Settings for the Freescall/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families.....	68
Processor-Specific Configuration Settings (Freescall/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Families).....	69
Infineon TriCore Family.....	71
Basics on the Infineon TriCore Family.....	71
Generic Configuration Settings for the Infineon TriCore Target Processor Family.....	71
Processor-Specific Configuration Settings (Infineon TriCore Family).....	73
Additional Processor-Specific Information (Infineon TriCore Family).....	75

Infineon XC2000 Family.....	76
Basics on the Infineon XC2000 Family.....	76
Generic Configuration Settings for the Infineon XC2000 Target Processor Family.....	76
Renesas M32R Family.....	78
Basics on the Renesas M32R Family.....	78
Generic Configuration Settings for the Renesas M32R Target Processor Family.....	78
Processor-Specific Configuration Settings (Renesas M32R Family).....	79
Renesas RH850 Family.....	81
Basics on the Renesas RH850 Family.....	81
Generic Configuration Settings for the Renesas RH850 Target Processor Family.....	81
Processor-Specific Configuration Settings (Renesas RH850 Family).....	82
Renesas SH2 Family.....	85
Basics on the Renesas SH2 Family.....	85
Generic Configuration Settings for the Renesas SH2 Target Processor Family.....	85
Processor-Specific Configuration Settings (Renesas SH2 Family).....	86
Renesas SH2A/SH4A Family.....	88
Basics on the Renesas SH2A/SH4A Family.....	88
Generic Configuration Settings for the Renesas SH2A/SH4A Target Processor Family.....	88
Processor-Specific Configuration Settings (Renesas SH2A/SH4A Family).....	89
Renesas V850E/V850E1 Family.....	91
Basics on the Renesas V850E/V850E1 Family.....	91
Generic Configuration Settings for the Renesas V850E/V850E1 Target Processor Family.....	91
Renesas V850E2 Family.....	93
Basics on the Renesas V850E2 Family.....	93
Generic Configuration Settings for the Renesas V850E2 Target Processor Family.....	93
Processor-Specific Configuration Settings (Renesas V850E2 Family).....	94
Texas Instruments TMS570 Family.....	96
Basics on the Texas Instruments TMS570 Family.....	96
Generic Configuration Settings for the Texas Instruments TMS570 Target Processor Family.....	96

Toshiba TMPV770 Family.....	98
Basics on the Toshiba TMPV770 Family.....	98
Generic Configuration Settings for the Toshiba TMPV770 Target Processor Family.....	98
 Preparing a Support Request	 101
Creating a Support Report.....	101
 Index	 103

About This Application Note

Contents

This document describes:

- Initial steps in connecting the DCI-GSI2 to an ECU.
- How to configure the DCI-GSI2 and optimize its configuration via the DCI Configuration Tool.
- How to adapt an A2L file to use the DCI-GSI2 with calibration, measurement or bypass tools.
- How to integrate the dSPACE Calibration and Bypassing Service, configure the DCI-GSI2, and set up a bypass.





This document is an application note and is meant as a quick starting guide for DCI-GSI2 users.



Note

This document describes the current status of the DCI-GSI2 and the DCI Configuration Tool. Both this document and the DCI-GSI2 functionality may be changed without notice. The features and functionalities of the DCI-GSI2 will be extended in the future, which may have an impact on the configuration and configuration procedures.

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.

Symbol	Description
Note	Indicates important information that you should take into account to avoid malfunctions.
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.

Safety Precautions

Introduction

To avoid risk of injury and/or damage to the dSPACE hardware, read and ensure that you comply with the following safety precautions. These precautions must be observed during all phases of system operation.

Warning About Using a DCI-GSI2

Introduction

Note the following warning and safety precautions when using a DCI-GSI2.

Danger potential

Connecting a DCI-GSI2 to an electronic control unit can affect system behavior. This can lead to unexpected or critical situations, or even constitute a risk of death. Therefore, only persons who are qualified to use a DCI-GSI2, and who have been informed of the dangers and possible consequences, are permitted to use the DCI-GSI2.

Before integrating the DCI-GSI2 and starting operation, read the warnings in this document carefully.

⚠ WARNING

Risk of serious injury or death due to electrical shock

The DCI-GSI2 is designed to be connected to devices that do not transmit hazardous voltages. According to the EN 61010 standard, a voltage higher than $33 V_{RMS}$ / $46.7 V_{PEAK}$ AC and 70 V DC is classified as hazardous. It constitutes a risk of serious injury or even death.

Make sure that your system provides safety provisions so that no hazardous voltages are applied to the DCI-GSI2, even in the event of electrical faults.

If there is a risk of hazardous voltages being applied to a DCI-GSI2, for example, when it is connected to an engine ECU which typically generates transient hazardous voltages for ignition, one of the following measures must be taken **to avoid the risk of serious injury or death due to electrical shock**:

- The DCI-GSI2 and all devices connected to it must be within a separate test area according to the locally valid safety standards for the installation and operation of electrical test equipment.
- dSPACE provides dedicated interface cables to ensure an electrically safe connection to the host PC for systems featuring voltages up to 300 V DC/AC_{RMS} , or 600 V_{peak}: The ETH_CAB2 Ethernet Connection Cable must be used for connecting the DCI-GSI2 to the host PC.
The DCI-GSI2 and the devices connected to it must be within a separate test area. When the above-mentioned cable is used, the host PC can be located outside the test area.

Using the DCI-GSI2 on wet locations

According to IEC 61010-1 (product safety), the DCI-GSI2 is not intended to be used on wet locations.

Unless the DCI-GSI2 is protected by a waterproof enclosure complying with the IP66 protection classification, using it in wet conditions might result in electric shock due to hazardous voltages or might damage the DCI-GSI2 and any connected ECU.

Electromagnetic compatibility

The DCI-GSI2 is a CE class A device. This equipment may cause interference in a residential installation. In this case the user is encouraged to perform appropriate measures to correct the interference. For details on CE compliance, refer to [Certifications of the DCI-GSI2 \(DCI-GSI2 Feature Reference !\[\]\(fe3aebe81acea8d45108cd2768939da7_img.jpg\)](#)).

Liability

It is your responsibility to adhere to instructions and warnings. Any unskilled operation or other improper use of this product in violation of the respective safety instructions, warnings, or other instructions contained in the user documentation constitutes contributory negligence, which may lead to a

limitation of liability by dSPACE GmbH, its representatives, agents and regional dSPACE companies, to the point of total exclusion, as the case may be. Any exclusion or limitation of liability according to other applicable regulations, individual agreements, and applicable general terms and conditions remain unaffected.

Preparing the DCI-GSI2

Introduction

This section describes initial steps in connecting a DCI-GSI2 to an ECU and gives instructions on how to configure a DCI-GSI2 via the DCI Configuration Tool.

Where to go from here

Information in this section

Initial Setup of the DCI-GSI2.....	16
Configuring the DCI-GSI2.....	18

Initial Setup of the DCI-GSI2

Introduction

This section describes initial steps in connecting a DCI-GSI2 to an ECU. If you are connecting a DCI-GSI2 to an ECU for the first time, you should follow the steps described in the following workflow.

Workflow for Initial Setup of the DCI-GSI2

Workflow

The following guidelines describe some basic steps in configuring your DCI-GSI2 and testing the connection of your DCI-GSI2 to your ECU:

1. Make sure the DCI Configuration Tool is installed on your PC. The tool is part of the DCI-GSI Configuration Package setup.
2. Make sure the DCI-GSI2 is not connected to your ECU.
3. Check if the Ethernet interface is correctly configured for your environment. The customer information leaflet which is shipped with every DCI-GSI2 states the Ethernet IP configuration with which your DCI-GSI2 was preconfigured by dSPACE. If necessary you have to change the Ethernet IP configuration. For details, refer to [Integrating a DCI-GSI2 into a Network \(ECU Interfaces Hardware Installation and Configuration !\[\]\(a9a7cf821bf949be41db724492f295be_img.jpg\)](#)).
4. Check if the DCI-GSI2 is configured for your ECU. The customer information leaflet which is shipped with every DCI-GSI2 states the target processor and interface type for which your DCI-GSI2 was preconfigured by dSPACE. You must not connect the DCI-GSI2 to your ECU unless it is configured for the target processor and interface type of your ECU.
5. Connect the DCI-GSI2 to the power supply and connect its Ethernet cable to your PC. For details, refer to [Connecting an ECU with DCI-GSI2 \(ECU Interfaces Hardware Installation and Configuration !\[\]\(2c367d84c99049f9805eec6142b5bc5d_img.jpg\)](#)).
6. Go through the configuration options and prepare an initial configuration. For detailed description of the configuration options, refer to [Main Window \(DCI-GSI2\) \(DCI Configuration !\[\]\(4d6500d074b243a4043016fae28892e4_img.jpg\)](#)).
7. If you modified any configuration parameters in the previous step, you have to download the device configuration to the DCI-GSI2. For instructions on how to do that, refer to [How to Download Configuration Files to a DCI-GSI2 \(DCI Configuration !\[\]\(96ebb722ad5476cd70154c5926aad591_img.jpg\)](#)).
8. Check that the DCI-GSI2 is in a consistent configuration status. Refer to [How to Check the Current Status and Configuration of a DCI-GSI2 \(DCI Configuration !\[\]\(2da405ec9c9a12d2adbd80327dcc4669_img.jpg\)](#)).
9. Export the interface description data (IF_DATA) of your current DCI-GSI2 configuration into your A2L file. This information is required by calibration, measurement and bypassing tools to access the ECU via the DCI-GSI2. For a detailed description of the workflow, refer to [Preparing the A2L File for the DCI-GSI2](#) on page 21.

10. Now you can connect the serial interface of the DCI-GSI2 to your ECU, and the DCI-GSI2 is ready for use.

Related topics**Basics**

[Basics on Configuring a DCI-GSI2 Using the DCI Configuration Tool.....](#) 18

Configuring the DCI-GSI2

Introduction

Gives information on configuring and optimizing the DCI-GSI2 configuration settings.

Where to go from here

Information in this section

Basics on Configuring a DCI-GSI2 Using the DCI Configuration Tool..... 18

Gives you information on configuring the DCI-GSI2 with the DCI Configuration Tool.

Configuring and Optimizing the DCI-GSI2 Settings..... 19

Once you have successfully connected your DCI-GSI2 to the ECU and thus ensured there is no hardware misconfiguration, you should go through the configuration settings once more for optimization. Optimizing the configuration settings will ensure that your system achieves maximum performance when performing measurements and bypassing.

Basics on Configuring a DCI-GSI2 Using the DCI Configuration Tool

Introduction

Configuring a DCI-GSI2 means checking and changing the device configuration, performing firmware updates for a DCI-GSI2, and adapting A2L files.

Basics

The DCI Configuration Tool can be used to perform the following steps:

- Download new firmware to a DCI-GSI2. Refer to [How to Download New Firmware to a DCI-GSI2 \(DCI Configuration !\[\]\(5a0d662075632df1b39c9e3427a70093_img.jpg\)](#)).
- Download configuration files to a DCI-GSI2. Refer to [How to Download Configuration Files to a DCI-GSI2 \(DCI Configuration !\[\]\(b9aaeddcca3b0cfd727d0e19f8b22e6b_img.jpg\)](#)).
- Check the current status and configuration of a DCI-GSI2. Refer to [How to Check the Current Status and Configuration of a DCI-GSI2 \(DCI Configuration !\[\]\(7985cfc9ac20c5a67d1a49b8edd9370c_img.jpg\)](#)).
- Upload and store the configuration of a DCI-GSI2. Refer to [How to Upload and Store the Configuration of a DCI-GSI2 \(DCI Configuration !\[\]\(c3dc9c5d8504b4ff44583fa2a53f68d3_img.jpg\)](#)).
- Adapt an A2L file automatically. Refer to [How to Adapt an A2L File Automatically \(DCI Configuration !\[\]\(dcc90d5dff4daebc8e015e38f89e1f01_img.jpg\)](#)).
- Perform diagnostics tests to check and optimize the function of a DCI-GSI2. Refer to [How to Perform Start-Up Diagnostics on a DCI-GSI2 \(DCI Configuration !\[\]\(5a7dbb8a52a41ee78efaae6ec4edbab9_img.jpg\)](#)).

Note

You should modify the configuration of the DCI manually only if you are very familiar with device configuration.

Before changing the configuration of a DCI-GSI2, it is recommended to create a backup of it. The DCI Configuration Tool comes with a default configuration file (G2C file) for each target processor family available for the DCI-GSI2. You can find the G2C files in the

`DCI_ConfigurationTool\DCI-GSI2\Configuration` folder in your dSPACE DCI-GSI Configuration Package installation. Since dSPACE archives the configuration of a DCI-GSI2 before it is shipped, you can also get a backup of the initial configuration of your DCI-GSI2 from dSPACE, if necessary.

Related topics**Basics**

[Configuring and Optimizing the DCI-GSI2 Settings](#)..... 19

Configuring and Optimizing the DCI-GSI2 Settings

Introduction

Once you have successfully connected your DCI-GSI2 to the ECU and thus ensured there is no hardware misconfiguration, you should go through the configuration settings once more for optimization. Optimizing the configuration settings will ensure that your system achieves maximum performance when performing measurements and bypassing.

G2C configuration file

The DCI-GSI2 configuration is performed by editing a G2C configuration file. The file contains a valid XML format and can be edited in a text editor or any XML editor. However, it is recommended to use the DCI Configuration Tool to modify the configuration of a DCI-GSI2.

For detailed information on configuring a DCI-GSI2, refer to [Configuring a DCI-GSI2 \(DCI Configuration !\[\]\(b792654f2cef9719eabeb6c5be00811e_img.jpg\)](#)).

Tip

You can find a default G2C configuration file for each target processor family and target interface available for the DCI-GSI2 in the `DCI_ConfigurationTool\DCI-GSI2\Configuration` folder in your dSPACE DCI-GSI Configuration Package installation. Since dSPACE archives the configuration of a DCI-GSI2 before it is shipped, you can also get a backup of the initial configuration of your DCI-GSI2 from dSPACE, if necessary.

DCI-GSI2 configuration settings Detailed descriptions of the configuration settings relevant to configuring a DCI-GSI2 for your ECU are given in the user documentation of the dSPACE DCI Configuration Tool. Refer to [Main Window \(DCI-GSI2\) \(DCI Configuration !\[\]\(21199eb166cc97331a0c54c649195dcc_img.jpg\)](#)).

Related topics

Basics	
Basics on Configuring a DCI-GSI2 Using the DCI Configuration Tool.....	18

Preparing the A2L File for the DCI-GSI2

Introduction

This section provides detailed information on the preparations which have to be performed in the A2L file to allow a measurement, calibration or bypass tool to interpret the interface-specific parameters.

Basics on Preparing the A2L File for the DCI-GSI2

Introduction

To allow a measurement, calibration or bypassing tool to work with a DCI-GSI2, an A2L file is necessary. The A2L file must contain certain structures. There are different ways to integrate these structures into an A2L file.

Make sure that your A2L file conforms to the A2L specification. The latest ASAP2 version 1.6 can be downloaded from <http://www.asam.net>.

Adapting an existing A2L file automatically

The DCI Configuration Tool offers the option to automatically insert and/or modify the structures needed for a DCI-GSI2 into an existing A2L file. For more information, refer to [How to Adapt an A2L File Automatically \(DCI Configuration !\[\]\(8bba887393ca45b761e5cb49e755e762_img.jpg\)](#)).

Adapting an existing A2L file manually

Using a text editor, you can adapt an existing A2L file manually by inserting and modifying the necessary parts into the file. In general, you have to perform the following steps:

- Adapt the A2ML description
- Insert the IF_DATA XCP structure
- Insert the IF_DATA dSPACE_XCP structure
- Extend the MEMORY_SEGMENT structures with DATA program type by an IF_DATA XCP structure
- Insert the MEASUREMENT structures for the system variables

Tip

The IF_DATA XCPplus structure allows you to describe multiple instances of one and the same transport layer. It is recommended to use IF_DATA XCPplus when using the DCI-GSI2 with multiple tools in parallel.

Related topics

Basics

Adapting or Generating an A2L File..... 55

Setting Up a Bypass With a DCI-GSI2

Introduction	This section provides step-by-step instructions on setting up a bypass with a DCI-GSI2. It covers all the major steps from integrating the dSPACE Calibration and Bypassing Service, through configuring a DCI-GSI2 device, to adapting an A2L file and creating a bypass model.
---------------------	--

Where to go from here

Information in this section

Introduction to Setting Up a Bypass with a DCI-GSI2.....	24
Describes the ways to perform bypassing with a DCI-GSI2 and shows the workflow for setting up a service-based bypass.	
Integrating the ECU Service.....	28
To perform service-based bypassing, the dSPACE Calibration and Bypassing Service has to be integrated into the ECU software.	
Configuring the DCI-GSI2.....	47
After the dSPACE Calibration and Bypassing Service is integrated, you must configure the DCI-GSI2 device and the service parameters.	
Testing the Integrated Service.....	53
To check if the service integration was done correctly, the service functionality should be tested.	
A2L File Adaptation.....	55
The DCI-GSI2 is supported by the RTI Bypass Blockset as a standard XCP device with some additional proprietary features. The device information is transported via an A2L file, which can then be imported in the SETUP block of the blockset.	
Bypass Model Configuration.....	57
To perform bypassing with the DCI-GSI2, the RTI Bypass Blockset is used within MATLAB/Simulink to create the bypass model.	

Introduction to Setting Up a Bypass with a DCI-GSI2

Introduction	Describes the ways to perform bypassing with a DCI-GSI2 and shows the workflow for setting up a service-based bypass.
---------------------	---

Where to go from here

Information in this section

When to Use the DCI-GSI2.....	24
The DCI-GSI2 is one of several possible ECU interfaces. This topic shows which cases to use the DCI-GSI2 in.	
Bypassing With or Without an ECU Service.....	25
Bypassing with the DCI-GSI2 is performed by combining synchronous data acquisition (measurement of ECU variables) and synchronous data stimulation (modification of ECU variables). There are several options for bypassing with the DCI-GSI2.	
Workflow for Setting Up a Service-Based Bypass.....	26
There is a typical workflow you must follow when you want to set up a service-based bypass with the DCI-GSI2.	

When to Use the DCI-GSI2

Introduction	The DCI-GSI2 is one of several possible ECU interfaces. This topic shows which cases to use the DCI-GSI2 in.
---------------------	--

Choosing the ECU interface

The DCI-GSI2 is a high-performance interface for accessing the debug interfaces of modern ECU microcontrollers. These debug interfaces allow the DCI-GSI2 to perform real-time accesses to the ECU microcontroller memory, i.e., without halting the microcontroller.

A DCI-GSI2 is usually chosen for an ECU project when other interfaces, such as XCP on CAN, XCP on Ethernet or a DPMEM POD, are either not available or do not provide enough performance for a real-time function bypass.

Some ECU microcontrollers also provide a real-time trace interface. Such an interface transmits program flow or data access information in real time to an attached debug tool via a dedicated trace interface. The DCI-GSI2 can be used with such trace interfaces to perform data measurement or function bypassing without having to integrate a service into the ECU software.

Related topics

Basics

Bypassing With or Without an ECU Service.....	25
Workflow for Setting Up a Service-Based Bypass.....	26

Bypassing With or Without an ECU Service

Introduction

Bypassing with the DCI-GSI2 is performed by combining synchronous data acquisition (measurement of ECU variables) and synchronous data stimulation (modification of ECU variables). There are several options for bypassing with the DCI-GSI2.

Choosing the bypass method

The bypass method that you chose depends on the capabilities of the ECU microcontroller hardware and the requirements on data consistency. The following table shows the bypass methods that are possible with a DCI-GSI2, including their advantages and disadvantages.

	Bypassing Without Changes to ECU Software	Bypassing With Changes to ECU Software
With Data Consistency	<p>Using <i>Fast Variable Overwrite</i></p> <ul style="list-style-type: none"> Requires access to microcontroller trace interface. There are certain timing restrictions. Works only with a limited amount of write variables. 	<p>Using <i>dSPACE Calibration and Bypassing Service</i></p> <ul style="list-style-type: none"> Up to 65,535 bypass events based on a single hardware event can be integrated. Up to 255 events can be used in parallel for reading and writing global A2L variables. Extended bypassing mechanisms (double buffer mechanism, wait mechanism, failure checking mechanism) can be used. Integrated bypass status detection in ECU software
Without Data Consistency	<p>Asynchronously to the ECU, using oversampling</p> <ul style="list-style-type: none"> Uses DCI-GSI2 internal timer (1 ms, 10 ms, etc.). Data is written very fast and periodically to the ECU. <p>Synchronously to the ECU</p> <ul style="list-style-type: none"> Uses available hardware events (e.g., BRKOUT/EVTO watchpoint pin, event polling). Depending on target microcontroller, only limited available amount of events. ECU timing needs to be known. 	<p>With <i>dSPACE Code Patch</i></p> <ul style="list-style-type: none"> Up to 65,535 bypass events based on a single hardware event can be integrated. Up to 255 events can be used in parallel for reading and writing global A2L variables. Minimal memory consumption and run time overhead.


The common case: service-based bypassing

The trace interfaces of ECU microcontrollers have very high electrical requirements due to the high bandwidth that is necessary to output all the microcontroller state information in real time. For this reason, trace interfaces are most often not available for bypassing applications.

Another frequent requirement is data consistency, because the goal of the bypass is to exchange control algorithm functions transparently, i.e., without influencing any other functionalities of the ECU software. Without a trace interface, this is usually possible only when an ECU service is used to read and write ECU variables. In addition, the ECU service provides failure detection, failure handling, and bypass status information, and also supports a high amount of bypass hooks.

dSPACE Calibration and Bypassing Service

The dSPACE Calibration and Bypassing Service consists of a set of source files that need to be integrated into the ECU software development project. The service is configured with defines and macros in a single C header file. Some functionalities can be adapted to the ECU requirements via a separate customization layer.

For further information on how the service works and a detailed description of all the functions and parameters, refer to [dSPACE Calibration and Bypassing Service Implementation](#) .

Related topics

Basics

When to Use the DCI-GSI2.....	24
Workflow for Setting Up a Service-Based Bypass.....	26

Workflow for Setting Up a Service-Based Bypass

Introduction

There is a typical workflow you must follow when you want to set up a service-based bypass with the DCI-GSI2.

Workflow

You have to perform the following steps to set up a bypass with the dSPACE Calibration and Bypassing Service and the DCI-GSI2:

1. Integrating the dSPACE Calibration and Bypassing Service into the ECU software.
Refer to [Integrating the ECU Service](#) on page 28.
2. Configuring the DCI-GSI2 for the bypass.
Refer to [Configuring the DCI-GSI2](#) on page 47.

- 3. (Optional) Testing the functionality of the integrated service.
Refer to [Testing the Integrated Service](#) on page 53.
- 4. Adapting the A2L file for bypassing.
Refer to [A2L File Adaptation](#) on page 55.
- 5. Creating a bypass model in MATLAB/Simulink.
Refer to [Bypass Model Configuration](#) on page 57.

Related topics

Basics

Bypassing With or Without an ECU Service.....	25
When to Use the DCI-GSI2.....	24

Integrating the ECU Service

Introduction

To perform service-based bypassing, the *dSPACE Calibration and Bypassing Service* has to be integrated into the ECU software. There are two ways to do this:

- Manual integration into the ECU source code (pre-build)
- Integration into the ECU binary code via the ECU Interface Manager (post-build)

Where to go from here

Information in this section

[Manual Integration of the Source Code.....28](#)

You can manually integrate the code of the service into the ECU source code (pre-build).

[Service Integration With the ECU Interface Manager.....41](#)

With the ECU Interface Manager, the code of the service is integrated into the ECU binary code (post-build). No ECU source code modification is required.

Manual Integration of the Source Code

Introduction

You can manually integrate the code of the service into the ECU source code (pre-build).

Where to go from here

Information in this section

[Basics on Manual Source Code Integration.....29](#)

Shows the basic workflow for manually integrating the service into the ECU source code, and provides arguments for and against manual service integration.

[Obtaining the Service Source Code.....30](#)

Provides information on where to find the installation file of the service.

[Integrating the Sources Into the ECU Software Project.....31](#)

You must copy the service source files into your ECU software project.

[Configuring the Service Parameters.....33](#)

The service consists of a single codebase that is used for a variety of ECU microcontroller architectures. It is therefore necessary to adapt it to the requirements of both the microcontroller and the compiler tool chain,

and to place the tool RAM that is used by the dSPACE Calibration and Bypassing Service..

Customizing the Service..... 38

Depending on the options used, customizing the service might be necessary.

Integrating the Service Functions..... 39

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

Basics on Manual Source Code Integration

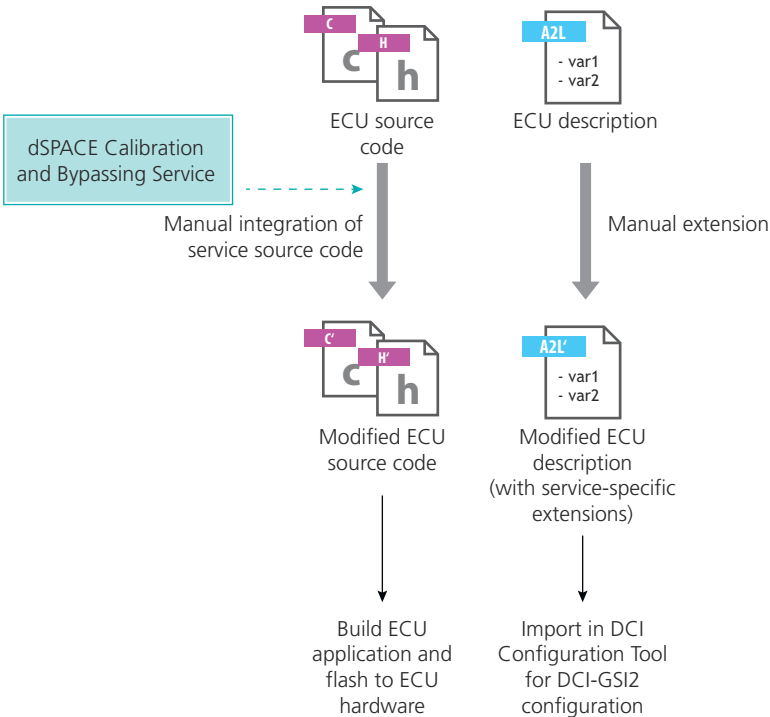
Introduction

There are arguments for and against manually integrating the ECU service into the ECU software.

Service integration into the ECU source code

You can integrate the code of the dSPACE Calibration and Bypassing Service into the ECU source code manually. This requires access to the ECU source code.

The following illustration shows the basic workflow for manually integrating the service into the ECU source code:



Note

There is an alternative approach to manually extending the A2L file with service parameters and importing the modified A2L file in the DCI Configuration Tool (as shown in the illustration above). The service parameters can also be configured manually in the DCI-GSI2 device configuration with the DCI Configuration Tool. Refer to [Manual Service Parameter Configuration](#) on page 48.

Advantages

Manually integrating the ECU service into the ECU software offers different advantages. The most important ones are listed below:

- The source code is provided by dSPACE free of charge.
- All the service parameters can be modified and adjusted to their optimal values for the desired ECU bypass scenario.
- The customization layer allows ECU-specific implementation of time checking routines and hardware trigger generation methods.

Disadvantages

Manually integrating the ECU service into the ECU software has the following disadvantages:

- Access to the ECU software source code and to a tool chain for building the ECU software are required.
- Service calls cannot be added or deleted after the ECU software has been created.
- You must be familiar with the concepts of the ECU service.

Related topics

Basics

Configuring the Service Parameters.....	33
Customizing the Service.....	38
Integrating the Service Functions.....	39
Integrating the Sources Into the ECU Software Project.....	31
Obtaining the Service Source Code.....	30

Obtaining the Service Source Code

Introduction

The dSPACE Calibration and Bypassing Service is part of the installation of ECU Interface Software. This section tells you where to find the installation file of the service.

Where to find the service installation file

You can install the source files and the service documentation by executing the setup file `dSPACECalibrationAndBypassingService_2.3.0.exe`, or a newer version if available. After you install and decrypt ECU Interface Software, you will find the `dSPACECalibrationAndBypassingService_<version>.exe` file in the `%ProgramData%\dSPACE\<InstallationGUID%\dsECU\Services` folder.

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

You can also acquire the service setup file from dSPACE directly without installing any other dSPACE software. To do so, contact your local dSPACE sales representative.

Related topics**Basics**

Basics on Manual Source Code Integration.....	29
Configuring the Service Parameters.....	33
Customizing the Service.....	38
Integrating the Service Functions.....	39
Integrating the Sources Into the ECU Software Project.....	31

Integrating the Sources Into the ECU Software Project

Introduction

You must copy the service source files into your ECU software project.

Service layout

After installing the service setup you will find several header and code files in the following folder structure below the `<ServiceInstallationFolder>` folder:

Subfolder	Service Files	Description
\Doc	dSPACECalibrationAndBypassingServiceFeatures.pdf	Contains a feature reference of the service.
	dSPACECalibrationAndBypassingServiceImplementation.pdf	Contains detailed information on the service functionalities, functions, and parameters and how to integrate the service into an ECU application.

Subfolder	Service Files	Description
\Source	<ul style="list-style-type: none"> ▪ dsECUboot.c ▪ dsECUboot.h ▪ dsECUBOOTCFG.h 	Boot loader files for ECU flashing Note These files are not required for bypassing.
	<ul style="list-style-type: none"> ▪ dsECUcmd.c ▪ dsECUcmd.h ▪ dsECUcmd_include.h 	Command service files for ECU flashing or ECU calibration Note These files are not required for bypassing, but must be present.
	dsECUCFG.h	Configuration header file containing the definitions of all the service parameters and target adaptation macros
	<ul style="list-style-type: none"> ▪ dsECUcustom.c ▪ dsECUcustom.h 	Customization layer files where the ECU time checking routines or hardware trigger method might need to be implemented
	<ul style="list-style-type: none"> ▪ dsECUSvc.c ▪ dsECUSvc.h 	Service source files Note These files do not need to be modified.

Integrating the source files into the ECU software project

To integrate the service source files into your ECU software project, copy all the source files, except for the boot loader files, into your ECU software project. Make sure that they will be compiled and linked to the ECU code.

Next step

Now you must adapt the `dsECUcustom.c` and `dsECUCFG.h` files to your needs. All other files do not need to be modified and can remain unchanged. Refer to [Configuring the Service Parameters](#) on page 33.

Related topics

Basics

Basics on Manual Source Code Integration.....	29
Configuring the Service Parameters.....	33
Customizing the Service.....	38
Integrating the Service Functions.....	39
Obtaining the Service Source Code.....	30

Configuring the Service Parameters

Introduction

The service consists of a single codebase that is used for a variety of ECU microcontroller architectures. It is therefore necessary to adapt it to the requirements of both the microcontroller and the compiler tool chain, and to place the tool RAM that is used by the dSPACE Calibration and Bypassing Service. This is done in the `dsECUCFG.h` header file.

All of the parameters and macros are documented via source code comments, but the most important ones are explained below.

Basic settings

All the data types used by the service are defined in the `dsECUCfg.h` header file. They all start with 'DSECU_'. You must adapt them to your ECU microcontroller architecture.

Note

For a common 32-bit microcontroller, the data types do not need to be modified.

DSECU_NUMBER_TOOLS This define configures the amount of service instances that will be provided by the dSPACE Calibration and Bypassing Service. The service supports a total of up to two instances. But for bypassing with the DCI-GSI2 only a single instance is needed, even if data acquisition is performed in parallel with the bypass. Therefore, this define should be set to 1.

DSECU_PROCESSOR_TYPE This define has to be set to the value according to the byte order of the ECU microcontroller. For little endian (Intel format) systems set this value to 0, for big endian (Motorola format) systems set it to 1.

Memory layout settings

DSECU_DPMEM_OFFSETADDR0 Defines the start address of the tool RAM, which is a memory area used as the interface between the dSPACE Calibration and Bypassing Service and the DCI-GSI2. The value can either be a fixed number or an expression that is evaluated at compile time.

DSECU_DPMEM_SIZE0 Defines the size of the tool RAM in bytes. The value can either be a fixed number or an expression that is evaluated at compile time. The necessary size of the tool RAM depends on several parameters, but most importantly on the amount of READ and WRITE blocks in the bypass model and the amount of variables that will be read/written by the service in total. For examples of typical bypass scenarios and their service configuration settings, see [Typical bypass scenarios](#) on page 37.

Note

The tool RAM is used exclusively by the service and must not be accessed by any other ECU code. It should therefore be marked as reserved in your compiler tool chain. In addition, accesses to the tool RAM area must be uncached. In most cases, this can be configured in the Memory Management Unit of the ECU microcontroller, or by choosing an appropriate address mapping for the tool RAM.

Other basic parameters

DSECU_SUBINT_TYPE For triggering the DCI-GSI2 whenever the service has sampled DAQ data, the subinterrupt mechanism is used. It allows generating multiple triggers when only one hardware event can be generated by the ECU, for example, by using an event out pin.

The DCI-GSI2 supports two subinterrupt types:

- **DSECU_SUBINT_1_BIT**: The bit-based subinterrupt mechanism requires the minimum amount of memory. This also reduces the bypass turnaround time because less data needs to be read by the DCI-GSI2. However, if service calls can be interrupted by each other it is necessary to implement the interrupt enable/disable macros (**DSECU_INT_...**) to avoid the potential loss of triggers.
- **DSECU_SUBINT_8_BIT**: The byte-based subinterrupt mechanism requires one byte per subinterrupt. It is less efficient, but it does not require temporary interrupt disabling.

DSECU_SINT_NUMBER Defines the maximum amount of subinterrupts supported by the service. The higher this number, the more memory is required by the service, and the bypassing latency also increases slightly. This value should be larger than or equal to the maximum amount of concurrently active event channels that perform read operations, e.g., READ or INTERRUPT blocks in the bypass model or DAQ events in ControlDesk.

DSECU_MAX_SERVICE_COUNT Defines the maximum amount of distinct service calls that can be integrated into the ECU application. This parameter should be set to a value greater than 255 (the default value) only if necessary.

Timeout detection

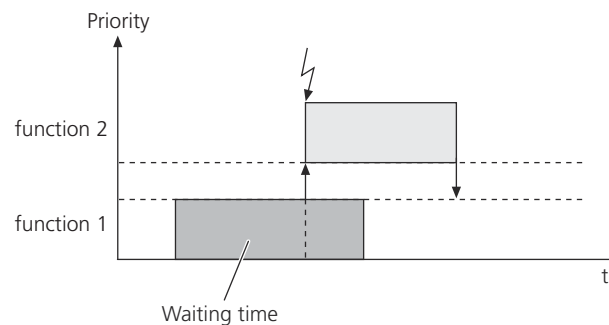
The dSPACE Calibration and Bypassing Service supports the wait mechanism, which enables the ECU to wait for a valid response from the prototyping system. If the ECU does not receive any new data until a predefined timeout, the ECU continues working in a defined way.

There are two mechanisms that can be used to detect a timeout in the wait mechanism:

- *Timeout detection via custom timeout mechanism* (proposed mechanism): A custom time stamp functionality is used to detect the timeout.

To specify the custom timeout mechanism, the `DSECU_TIMEOUT_CUSTOM_FUNCTION` and `DSECU_TIMEOUT_TIMESTAMP_TYPE` defines are relevant. For more details on the defines and the relevant functions, refer to [Customizing the Service](#) on page 38.

With the custom timeout mechanism enabled, the waiting process of a function is not interrupted by the execution of another function, but continues simultaneously. This means that 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, as shown by the example in the following illustration.

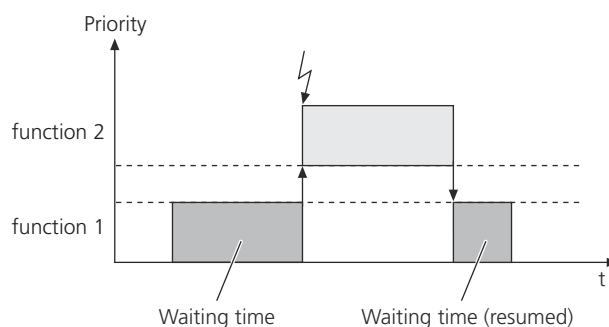


- **Timeout detection via calibrated wait loop:** The timeout is implemented as a wait loop in the dSPACE Calibration and Bypassing Service. The service executes a simple instruction sequence continuously. The number of repetitions is calculated by a configured timeout time and a scaling factor. To specify timeout detection via a calibrated wait loop, the following configuration options are required:

DSECU_TIMEOUT_CUSTOM_FUNCTION: This define enables or disables the custom time stamp functionality for specifying the timeout. It must be set to `DSECU_DISABLED`.

DSECU_TIMEOUT_SCALING_10US: This define configures a timeout scaling factor. It specifies the number of wait loops that the microcontroller performs in a time of 10 μ s and therefore provides the base unit for the timeout in the wait mechanism. The timeout expires when the maximum number of wait loops is reached. The maximum number of loops is determined by the timeout time specified in the ECU event configuration, multiplied with 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.

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, as shown in the following illustration.



DAQ and bypass options

DSECU_DAQ_BYPASS_SERVICE This define enables the DAQ capabilities. It has to be set to DSECU_ENABLED for bypassing and data acquisition.

DSECU_BYPASS This define enables the STIM capabilities. It has to be set to DSECU_ENABLED for bypassing.

DSECU_CUSTOM_HW_TRIGGER The setting of this parameter depends on the type of triggering that is supported by the ECU microcontroller and the connection to the DCI-GSI2. If this parameter must be set to DSECU_ENABLED, the `dsecu_custom_trigger_hw_int()` function in the `dsECUcustom.c` file must be implemented accordingly.

The following table lists the supported cases:

Trigger Type	Supporting Microcontroller Families	DSECU_CUSTOM_HW_TRIGGER Setting
ECU pin	All	DSECU_ENABLED The triggering of the GPIO pin that is connected to the EVTO pin of the DCI-GSI2 must be implemented in the <code>dsecu_custom_trigger_hw_int()</code> function.
Watchpoint pin	<ul style="list-style-type: none"> Infineon TriCore Infineon XC2000 Renesas V850 Renesas RH850 Renesas M32R Freescale/NXP MPC5xxx STMicroelectronics SPC5xxx 	DSECU_DISABLED
Watchpoint message	<ul style="list-style-type: none"> Freescale/NXP MPC5xxx STMicroelectronics SPC5xxx Renesas V850E2 	DSECU_DISABLED
Watchpoint polling	<ul style="list-style-type: none"> Infineon TriCore AURIX Infineon TriCore AURIX 2G 	DSECU_DISABLED
Poll event	<ul style="list-style-type: none"> Infineon TriCore Renesas V850E2 Renesas RH850 Renesas SH2A/SH4A Freescale/NXP MPC57xx STMicroelectronics SPC57xx/SPC58xxx 	DSECU_ENABLED Setting the appropriate bits of the event polling register must be implemented in the <code>dsecu_custom_trigger_hw_int()</code> function.

Trigger Type	Supporting Microcontroller Families	DSECU_CUSTOM_HW_TRIGGER Setting
Data trace event	<ul style="list-style-type: none"> ▪ Freescale/NXP MPC5xxx ▪ STMicroelectronics SPC5xxx ▪ Renesas V850E2 	DSECU_DISABLED

DSECU_DOUBLE_BUFFER If this parameter is set to DSECU_ENABLED, a double buffer mechanism can be activated by the bypass system. The double buffer mechanism ensures that only consistent data sets are written to the ECU memory. In most cases this feature should be enabled.

DSECU_FAILURE_CHECKING Support for the failure checking mechanism is enabled when this parameter is set to DSECU_ENABLED. It allows the service to detect whether new data has been received from the RCP system since the service was last called. The double buffer mechanism must be enabled for this setting to take effect.

DSECU_ECU_WAITS Support for the wait mechanism is enabled when this parameter is set to DSECU_ENABLED. It allows the service to wait for new data from the RCP system for a configurable amount of time. The double buffer mechanism must be enabled for this setting to take effect.

Either the custom timeout detection functions need to be implemented, or the DSECU_TIMEOUT_SCALING_10US parameter (see above) must be configured correctly.

Typical bypass scenarios

To assist you in determining the optimal tool RAM size, the following table lists some example bypass scenarios. It is assumed that the dSPACE Calibration and Bypassing Service is configured to support up to 255 service instances and uses the bit-based subinterrupt mechanism with 31 subinterrupts. It is also assumed that the double buffer mechanism is enabled for all write variables.

Number of Used Events	Number of 32-Bit Variables ...		Tool RAM Size
	Read	Written	
4	61	8	1 KB
8	120	32	2 KB
16	261	64	4 KB
32	544	128	8 KB

Related topics

Basics

Basics on Manual Source Code Integration.....	29
Customizing the Service.....	38
Integrating the Service Functions.....	39
Integrating the Sources Into the ECU Software Project.....	31
Obtaining the Service Source Code.....	30

Customizing the Service

Introduction

Apart from the configuration parameters in the `dsECUcfg.h` header file, the service needs only very few customizations. Depending on the options used, customization might not be necessary at all.

Custom event triggering

You can use the `dsecu_custom_trigger_hw_int()` function to implement custom mechanisms for triggering the DCI-GSI2. In general, this is necessary only if a general purpose I/O pin (GPIO pin) or the event polling mechanism is used.

The code for toggling a GPIO pin is specific to the ECU microcontroller and the pin that is used. The pin must be active low, which means that it generates a short low burst to generate a trigger. The signal burst should be low for at least 50 ns to ensure detection by the DCI-GSI2.

To generate a polling event, a bit has to be set in a special, microcontroller-specific register. When this polling mechanism is activated, the DCI-GSI2 will constantly read this register and detect any changes and trigger the according events. For example, if bit 0 is set in this register, the DCI-GSI2 will trigger the poll event 0.

The polling mechanism is supported by the following microcontroller families:

- Infineon TriCore
- Renesas V850E2
- Renesas RH850
- Renesas SH2A/SH4A
- Freescale/NXP MPC57xx
- STMicroelectronics SPC57xx/SPC58xx

For the name of the polling register for triggering polling events, refer to the hardware manual of your microcontroller.

Tip

For Infineon AURIX microcontrollers, it is recommended to enable the *trigger in protocol* option in the DCI-GSI2 configuration if the event polling mechanism is used. If you use the event polling mechanism, it can significantly reduce the latency penalty. Refer to [Target Interface Page \(DCI Configuration !\[\]\(ab4e2b3fc7e7887b7a72f548aa6f5e60_img.jpg\)](#)).

Custom timeout detection

You can use a custom timeout mechanism to detect a timeout in the wait mechanism.

DSECU_TIMEOUT_CUSTOM_FUNCTION This define enables or disables the custom time stamp functionality for specifying the timeout. It must be set to `DSECU_ENABLED`.

DSECU_TIMEOUT_TIMESTAMP_TYPE This parameter specifies the data type of the time stamp used with the custom timeout mechanism. The time stamp can have any data type.

In addition, the following two functions must be implemented. You can find them in the `dsECUcustom.c` file. They are called whenever the service wants to check if a timeout occurred.

- The `dsecu_custom_timeout_timestamp_get()` function is used to calculate a time stamp. The returned time stamp value will be used by `dsecu_custom_timeout_pending_check` for timeout calculation.
- The `dsecu_custom_timeout_pending_check()` function checks periodically whether a timeout occurred. For this, it uses the time stamps returned by the `dsecu_custom_timeout_timestamp_get` function.

Custom interrupt locking/unlocking

In a multithreaded environment, the service can be called multiple times at the same time by different tasks. This is no problem in general, but if the bit-based subinterrupt mechanism is used, triggers might get lost under certain circumstances. Therefore, if the bit-based subinterrupt mechanism is used, it is recommended to implement the interrupt locking mechanism. This is done by defining the following macros in the `dsECUcfg.h` file:

- `DSECU_INT_STATUS_VAR_DECL`
- `DSECU_INT_SAVE_AND_DISABLE`
- `DSECU_INT_RESTORE`

Related topics

Basics

Basics on Manual Source Code Integration.....	29
Configuring the Service Parameters.....	33
Integrating the Service Functions.....	39
Integrating the Sources Into the ECU Software Project.....	31
Obtaining the Service Source Code.....	30

Integrating the Service Functions

Introduction

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

Integrating the service functions

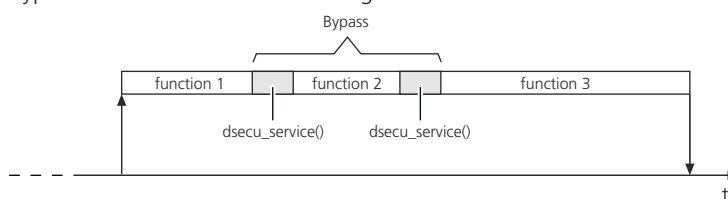
To integrate the dSPACE Calibration and Bypassing service into the ECU application it is only necessary to call the following three functions:

dsecu_service_init(0) The service is initialized by calling the `dsecu_service_init` function. The function must always be integrated into the ECU code. It has to be called once for each service instance used. For

bypassing with the DCI-GSI2, where only a single instance is needed, this means that the function is called only once. The function must be called prior to any other dSPACE Calibration and Bypassing Service function.

dsecu_service_background() The **dsecu_service_background** function must always be integrated into the ECU code. It performs routine work and should be called periodically in a low priority task. It should be called at least two times a second, but calling it faster might speed up the starting/stopping of a data acquisition or bypass.

dsecu_service(SvcId) The **dsecu_service** function is the actual service function that should be called whenever data is to be read or written by the service. For pure data acquisition, it is usually called at the end of functions or tasks. For bypassing scenarios, it is called before and after each function to be bypassed as shown in the following illustration.



The only parameter is a unique 16-bit 'service ID', which identifies each service call. Later on in the bypass model or in the measurement tool, the reading and writing of ECU variables can be assigned to the service IDs.

Note

If you work with a multicore ECU application, the service calls can be performed by independent processing cores if the following conditions are met:

- Each service ID must be used exclusively by one core only.
- When the application for each processing core is created in a separate compilation step so that the cores do not share the same address space, each core must call the **dsecu_service_init()** and **dsecu_service_background()** functions.
- The bit-based subinterrupt mechanism requires that a mutual exclusion of the cores is implemented with the **DSECU_INIT_x** macros. It is therefore recommended to use the byte-based subinterrupt mechanism in multicore scenarios.

Building the ECU application and generating an A2L file

After the service function calls are integrated in the ECU source code, you must build an ECU application by using your specific build environment. The ECU application now is prepared for bypassing and can be flashed to the ECU hardware.

In addition, you must generate an A2L file according to the integrated ECU service. The A2L file must contain an appropriate interface-specific **IF_DATA** entry. This information must comply with a particular data format. For details, refer to the [Interface Description Data Reference](#) . You can manually extend

the A2L file with service parameters and then import it in the DCI Configuration Tool to load the modified configuration to the DCI-GSI2. Refer to [Configuring the DCI-GSI2](#) on page 47.

Related topics

Basics

Basics on Manual Source Code Integration.....	29
Configuring the Service Parameters.....	33
Customizing the Service.....	38
Integrating the Sources Into the ECU Software Project.....	31
Obtaining the Service Source Code.....	30

Service Integration With the ECU Interface Manager

Introduction

With the ECU Interface Manager, the code of the service is integrated into the ECU binary code (post-build). No ECU source code modification is required.

Where to go from here

Information in this section

Basics on Service Integration via ECU Interface Manager.....	41
Shows the basic workflow for integrating the service into the ECU binary code, and provides arguments for and against service integration with the ECU Interface Manager.	
Integrating the Service Into the ECU Binary Code.....	43
The ECU Interface Manager lets you prepare the binary code of ECU applications for service-based bypassing.	

Basics on Service Integration via ECU Interface Manager

Introduction

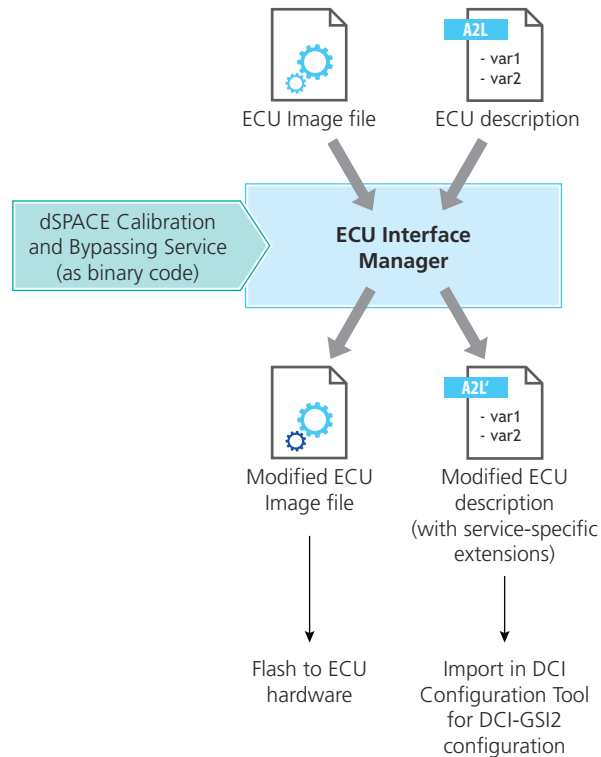
There are arguments for and against ECU service integration with the ECU Interface Manager.


Service integration via the ECU Interface Manager

The dSPACE ECU Interface Manager can analyze the binary code of an ECU application and visualize its program flow. In particular, it can identify conditional statements, function calls, and read/write accesses to global variables.

When the program flow is known, it is possible, among other things, to integrate the dSPACE Calibration and Bypassing Service into the ECU application to support external bypassing. This is done via precompiled *target software modules* that are provided by dSPACE.

The following illustration shows the basic workflow for integrating the service into the ECU binary code by using the ECU Interface Manager:



For more details on how the ECU Interface Manager works, what kind of information has to be provided for analyzing an ECU software binary, and all the features and workflows that are supported by it, refer to [ECU Interface Manager Manual](#) .

Advantages

The most important advantages of integrating the ECU service via the ECU Interface Manager are:

- The service can be integrated without any changes to the ECU source code.
- Service calls can be added and removed without having to recompile the ECU software.
- No deep knowledge on all the service configuration and customization options is required.

Disadvantages

Integrating the service source code via binary code modification has the following disadvantages:

- Some specific information to analyze an ECU binary is necessary.
- The environment must be able to apply the binary code mechanism. That is, code checking mechanisms must be disabled in the ECU application or CRC checksums must be recalculated.

Related topics**Basics**

[Integrating the Service Into the ECU Binary Code..... 43](#)

Integrating the Service Into the ECU Binary Code

Introduction

The ECU Interface Manager lets you prepare the binary code of ECU applications for service-based bypassing.

Integrating the service binary code

After the ECU application has been imported into an ECU Interface Manager project, the actual service integration is quite easy. You only have to import a *Target Software Module* package of the *dSPACE Calibration and Bypassing Service* into your project.

Integrating service function calls in the ECU binary code

After the service binary code is integrated, you must integrate service function calls for service initialization and data transfer to and from the ECU within your ECU application.

The following illustration shows some examples where service calls could be placed:

main()
task_scheduler()
<pre>task_reset() { initialization code of the ECU software... dsecu_service_init(); }</pre>
<pre>task_1ms() { some ECU code... dsecu_service(); functionX_to_bypass(); dsecu_service(); some ECU code... }</pre>
<pre>task_10ms() { some ECU code... dsecu_service(); functionY_to_bypass(); dsecu_service(); some ECU code... }</pre>
<pre>task_100ms() { some ECU code... dsecu_service_background(); }</pre>

Initializing the service To initialize the bypassing service, you have to add one init service call and also one background function call to the ECU application:

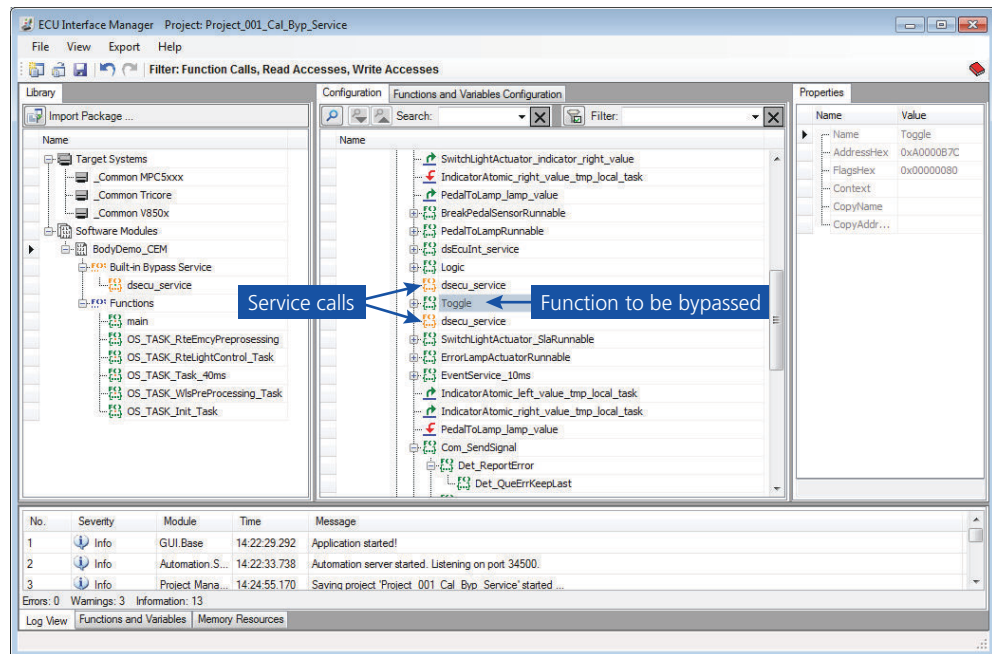
- The `dsecu_service_init()` initialization function must be called once initially before any other service function. You can place the init call at the beginning of a function that is executed only once, such as a main, a reset, or an init function or task.
- The `dsecu_service_background()` background function has to be called periodically several times a second. You can place the function call in a background task or a slowly running periodical task, e.g., a 100 ms task.

The ECU Interface Manager lets you add these service calls to the ECU application via drag & drop.

Inserting service function calls for the functions to be bypassed To prepare functions and variables for bypassing, the `dsecu_service()` function call must be placed at the beginning and at the end of each function that is to be bypassed.

The ECU Interface Manager provides several methods to insert the service calls to the ECU application. The most comfortable way is to use the Functions and Variables pane, which works with predefined behaviors for the insertion of service calls. After selecting the functions to be bypassed, the service calls above and below the functions are inserted automatically.

The following illustration shows the visualization of an inserted bypass hook in the ECU Interface Manager.



Note

If you work with a multicore ECU application, the service calls can be performed by independent processing cores if the following conditions are met:


- Each service ID must be used exclusively by one core only.
- The byte-based subinterrupt mechanism must be used.

Tip

The ECU Interface Manager can also be used to add `dsecu_service()` calls to an ECU application that already has the dSPACE Calibration and Bypassing Service integrated.

Exporting the ECU application and the A2L file

After the service calls for bypassing are integrated, you can export the modified ECU application to a new ECU Image file and a modified A2L file prepared for bypassing. The A2L file now contains information about the integrated service. If you do not want to merge the contents of the original A2L file and the bypass-

related information in a single new file, you can also specify output a separate A2L file with information on the service calls. Before the export is performed, you must configure it. The ECU Interface Manager has many configuration options for the export. For details, refer to [ECU Interface Manager Manual](#) .

The new binary file can then be flashed to the ECU. The exported A2L file can be imported to the DCI Configuration Tool in order to load the modified configuration to the DCI-GSI2 and to export the necessary XCP interface information into the A2L file. Refer to [Configuring the DCI-GSI2](#) on page 47.

Related topics

Basics

[Basics on Service Integration via ECU Interface Manager](#)..... 41

Configuring the DCI-GSI2

Introduction

After the dSPACE Calibration and Bypassing Service is integrated, you must configure the DCI-GSI2 device and the service parameters.

Where to go from here

Information in this section

[Basics on DCI-GSI2 Configuration](#).....47

The DCI-GSI2 provides a standardized XCP on Ethernet interface, which does not support the setting of arbitrary configuration parameters. It is therefore necessary to configure the device and the service parameters separately.

[Manual Service Parameter Configuration](#).....48

After you integrate the dSPACE Calibration and Bypassing Service manually into the ECU application, you usually do not have an A2L file containing the service configuration description. In this case, the service parameters can be configured manually with the DCI Configuration Tool.

[Service Parameter Configuration via A2L File](#).....51

The DCI-GSI2 can be configured for bypassing simply by importing an A2L file containing all the information required for bypassing via the DCI Configuration Tool.

Basics on DCI-GSI2 Configuration

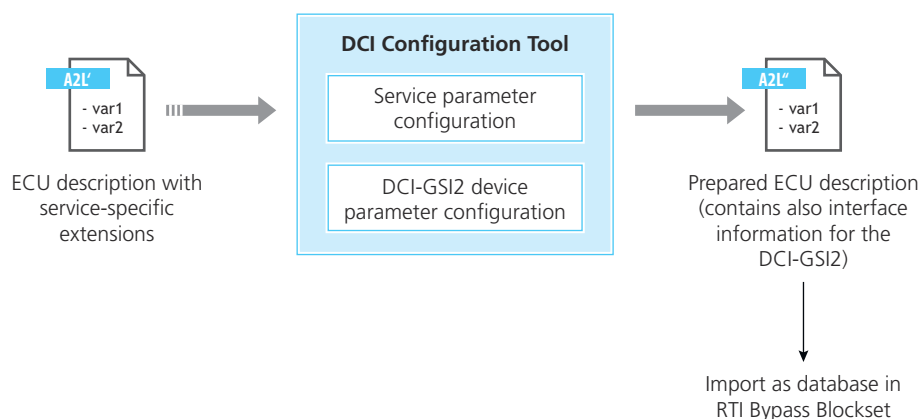
Introduction

The DCI-GSI2 provides a standardized XCP on Ethernet interface, which does not support the setting of arbitrary configuration parameters. It is therefore necessary to configure the device and the service parameters separately. This usually has to be done only once for each ECU software revision.

Configuration is done with the DCI Configuration Tool.

Configuring the DCI-GSI2

The following illustration shows the basic workflow for configuring the DCI-GSI2 for bypassing. It comes after either the workflow for manual service integration (see [Basics on Manual Source Code Integration](#) on page 29), or the workflow for service integration with the ECU Interface Manager (see [Basics on Service Integration via ECU Interface Manager](#) on page 41).



Basic device configuration

The DCI-GSI2 comes preconfigured with a default device configuration. Usually, the only basic configuration settings that need to be modified on the **Ethernet** page are the *Ethernet IP address* or the *DHCP configuration*.

For more details on the other basic configuration settings, refer to the [DCI Configuration](#) document.

ECU-specific device configuration

Depending on the ECU microcontroller and the debug interface that are used, some configuration settings on the **Target Interface** page might need to be adjusted. In most cases, the default configuration will work and sometimes only the **Interface clock** setting has to be adjusted. But for certain interface connection setups, some other technical parameters need to be modified. For more details, refer to the [DCI Configuration](#) document.

There are several other ECU-specific configuration parameters, but they are mostly required for setting up data calibration or ECU flash programming.

Related topics

Basics

Manual Service Parameter Configuration	48
Service Parameter Configuration via A2L File	51

Manual Service Parameter Configuration

Introduction

After you integrate the dSPACE Calibration and Bypassing Service manually into the ECU application, you usually do not have an A2L file containing the service configuration description. In this case, the service parameters can be configured manually with the DCI Configuration Tool.

Service parameters

First, you have to select the *dSPACE Calibration and Bypassing Service* on the Service Configuration page.

Then you have to configure all the service parameters on the General Parameters page. The ToolRAM address and/or ToolRam size might need to be taken from the ECU application MAP file. All the other service parameters can be taken from the `dsECUcfg.h` header file.

Tip

It is also possible to associate the service tool RAM address with an A2L variable so that the address is automatically updated whenever the A2L file changes with a new ECU software revision.

Service event trigger

The source of the service event trigger must be configured on the Event Trigger Source page. The values you have to configure on this page depend on the ECU microcontroller and debug interface that are used, and also on the way you implemented the custom event trigger, if you have done so.

If the `DSECU_CUSTOM_HW_TRIGGER` option is set to `DSECU_DISABLED`, the service generates a trigger by writing to the default trigger address. This address can be calculated by the following formula:

Trigger address = `DSECU_DPMEM_OFFSETADDR0` + `DSECU_DPMEM_SIZE` - 4

Note

If the `DSECU_CUSTOM_HW_TRIGGER` option is disabled, you can also set the Event source type to (*autodetect*). The DCI-GSI2 will then try to automatically determine the best event trigger source and configure the parameters accordingly.

The following table lists all the available options and the corresponding parameters:

Trigger Type	Supporting Microcontroller Families	Parameters
ECU pin	All	Parameter 1: Event pin number (usually 0)
Watchpoint pin	<ul style="list-style-type: none"> Infineon TriCore Infineon XC2000 Renesas V850 Renesas RH850 Renesas M32R Freescale/NXP MPC5xxx STMicroelectronics SPC5xxx 	Parameter 1: Trigger address Parameter 2: Trigger address extension (usually 0) Parameter 3: Event pin number (usually 0)
Watchpoint message	<ul style="list-style-type: none"> Freescale/NXP MPC5xxx STMicroelectronics SPC5xxx Renesas V850E2 	Parameter 1: Trigger address Parameter 2: Trigger address extension (usually 0)

Trigger Type	Supporting Microcontroller Families	Parameters
Watchpoint polling	<ul style="list-style-type: none"> Infineon TriCore AURIX Infineon TriCore AURIX 2G 	Parameter 1: Trigger address Parameter 2: Trigger address extension (usually 0)
Poll event	<ul style="list-style-type: none"> Infineon TriCore Renesas V850E2 Renesas RH850 Renesas SH2A/SH4A Freescale/NXP MPC57xx STMicroelectronics SPC57xx/SPC58xx 	Parameter 1: Poll register bit number (set to 0 if you write value 0x1 into the register)
Data trace write	<ul style="list-style-type: none"> Freescale/NXP MPC5xxx STMicroelectronics SPC5xxx Renesas V850E2 	Parameter 1: Trigger address Parameter 2: Trace unit (0 = <i>don't care</i>) Parameter 3: Trace access size (0 = <i>don't care</i>)
Timer ¹⁾	All	Parameter 1: Time unit (3 = 1 μ s, 6 = 1 ms, 9 = 1 s) Parameter 2: Time cycle (multiplier for time unit) Parameter 3: Timer offset [μ s] (only with time synchronization enabled)


¹⁾ The *Timer* trigger type does not use an event generated by the service in the ECU, but rather causes the DCI-GSI2 to poll the service trigger status cyclically in the specified time period. This causes delay and jitter, and should only be used if there is no other option.

Tip

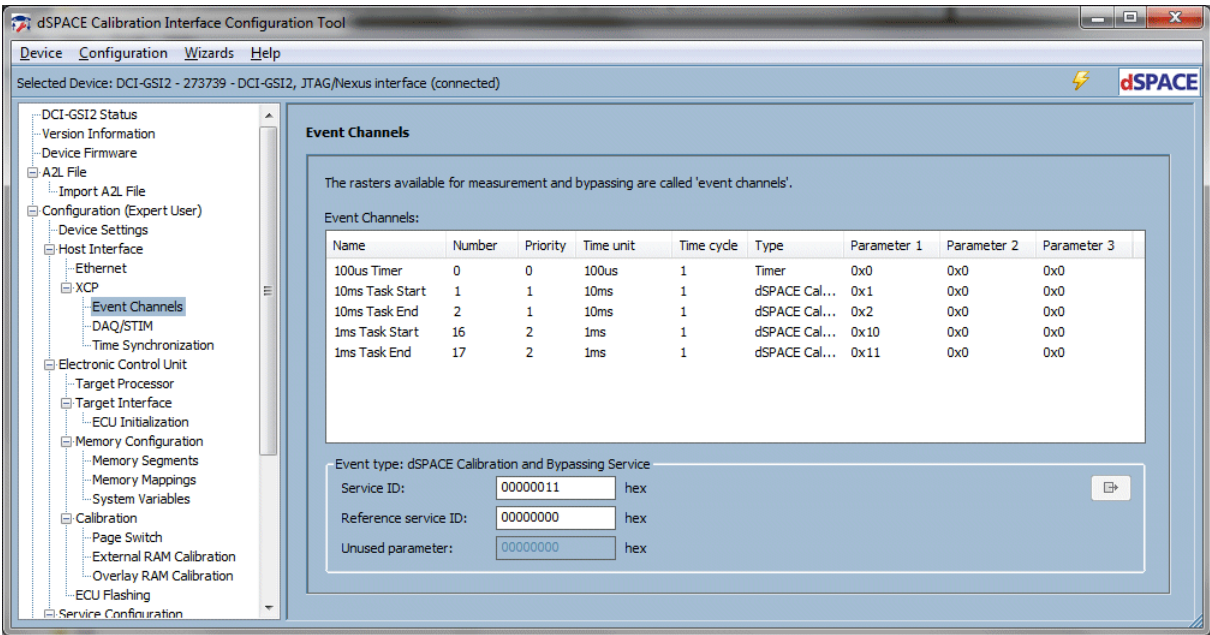
It is also possible to associate the first event trigger parameter with an A2L variable so that the address is automatically updated whenever the A2L file changes with a new ECU software revision.

Event channels

As the last step of the DCI-GSI2 configuration, the XCP event channels have to be configured on the Event Channels page. For each service call that has been integrated into the ECU application, an XCP event channel should be configured.

Each event channel gets a *name*, which will also appear in the A2L file and later allow you to assign ECU variable read and write accesses to ECU tasks via the RTI Bypass Blockset. A service event channel also has a unique *number*, a *priority* (0 is lowest, 255 is highest), a *time unit* and *time cycle*, the *type* dSPACE Calibration and Bypassing Service, and the service ID as the first parameter (*Parameter 1*). Optionally, another service ID can also be referenced with the second parameter, which is used for the buffer synchronization mechanism of the service. This mechanism allows the service to make sure that it is running without a sample step delay. For more details, refer to the [DCI Configuration](#)  document.

The following illustration shows the event channel configuration for a typical scenario with four service calls for bypassing two ECU tasks as an example:




Downloading the device configuration

After the event channels have been configured, make sure you download the device configuration to the DCI-GSI2. You can also save a copy of the configuration in a specific directory, if desired.

Related topics

Basics

Basics on DCI-GSI2 Configuration.....	47
Extended Bypassing Mechanisms (dSPACE Calibration and Bypassing Service Implementation )	
Service Parameter Configuration via A2L File.....	51

Service Parameter Configuration via A2L File

Introduction

After you integrate the dSPACE Calibration and Bypassing Service into the ECU application by using the ECU Interface Manager, you will also have exported an A2L file. This A2L file contains all the information about the service that is required for bypassing. Also in the case of manual service integration, you might have manually generated an A2L file containing the service configuration description. The DCI-GSI2 can therefore be configured for bypassing simply by importing the A2L file via the DCI Configuration Tool.

Importing the service information from the A2L file

After you have loaded the DCI-GSI2 configuration either by connecting to a DCI-GSI2 device or by loading it from a **G2C** file, you can import the service information from the A2L file on the Import A2L File page. Simply click Import A2L file and select the A2L file to be imported.

The service configuration parameters and the XCP event channels will be configured completely automatically. The DCI Configuration Tool will also try to configure the service event trigger, but in some cases the A2L file does not contain all the necessary information to do this fully automatically. For this reason, check the Event Trigger Source page after the A2L file import has completed and correct the settings, if necessary.

Note

Currently, the automatic configuration of the event trigger is not possible for the following event source types:

- Timer
- Watchpoint polling
- Data trace write
- Data trace write value

Related topics

Basics

Basics on DCI-GSI2 Configuration.....	47
Manual Service Parameter Configuration.....	48

Testing the Integrated Service

Introduction

To check if the service integration was done correctly, the service functionality should be tested.

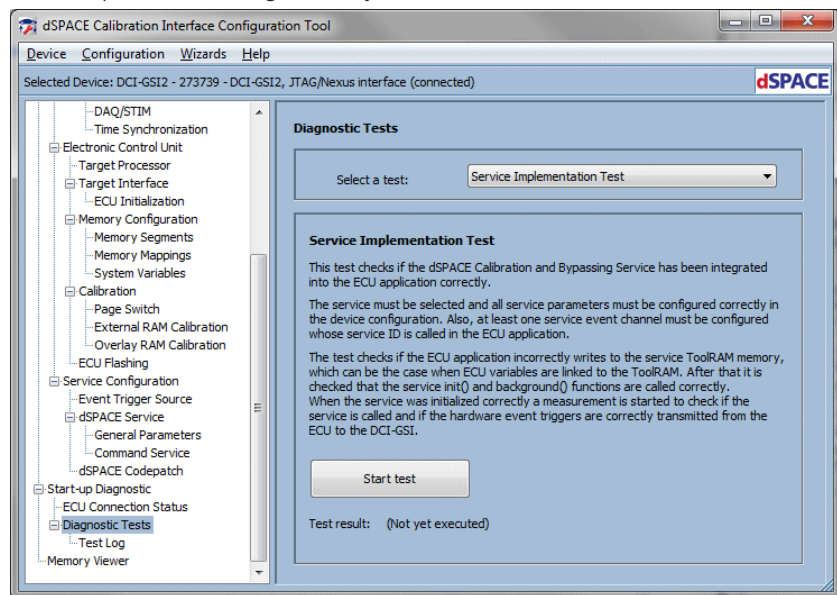
Testing the Service Functionality

Introduction

The service functionality can be tested by setting up a bypass and testing if everything works fine. However, the DCI Configuration Tool provides a simpler way to test the service functionality without having to set up and build a bypass model.

Testing the service implementation via DCI Configuration Tool

The DCI Configuration Tool lets you perform various diagnostic tests, one of which is the *Service Implementation Test*. Before the test can be performed, the service parameters have to be configured in the DCI-GSI2 configuration. For more details, refer to [Configuring the DCI-GSI2](#) on page 47. When you are done with the parameter configuration, you can start the test.



The DCI Configuration Tool performs several tests to verify that the service integration functions correctly. Among other things, the test checks the service function integration, the correct tool RAM placement, the exclusivity of accesses to the tool RAM by the service, and the correct function of the event trigger.

To check if all the integrated service calls are called correctly, you can also perform the *Event Channel Test*.

Related topics

HowTos

[How to Perform Start-Up Diagnostics on a DCI-GSI2 \(DCI Configuration !\[\]\(eafc244b53721dd1ec133f0772f70fc7_img.jpg\)\)](#)

A2L File Adaptation

Introduction

The DCI-GSI2 is supported by the RTI Bypass Blockset as a standard XCP device with some additional proprietary features. The device information is transported via an A2L file, which can then be imported in the SETUP block of the blockset.

Adapting or Generating an A2L File

Introduction

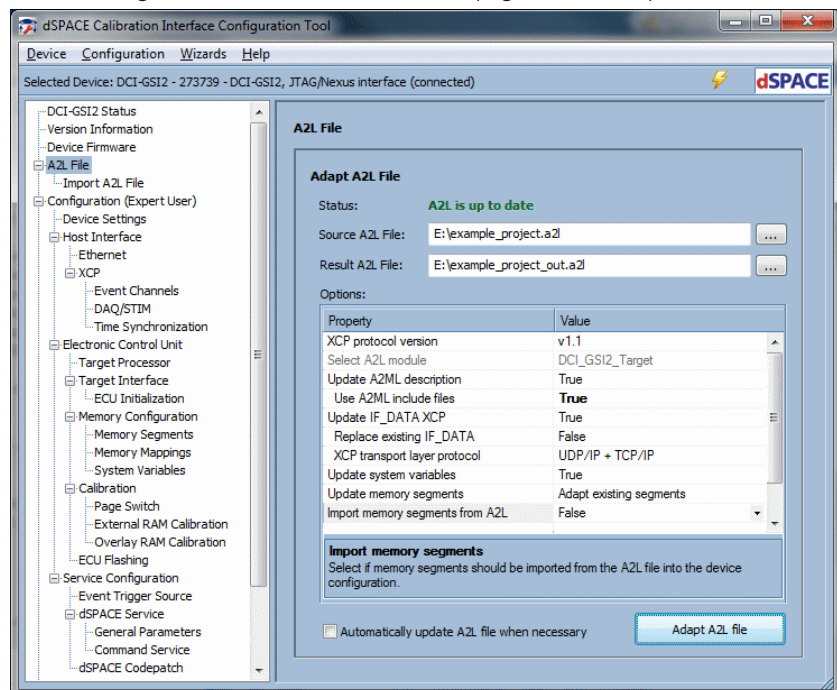
The adaptation of an existing A2L file or the generation of a new A2L file with just the interface information for the DCI-GSI2 can be done manually or automatically with the DCI Configuration Tool.

Generating or adapting an A2L file

The DCI Configuration Tool supports generating a new A2L file with all the necessary information for a bypass, measurement or calibration scenario with the DCI-GSI2. This is done on the A2L File page. You just need to specify the Result A2L File (and leave the Source A2L File field empty), select the appropriate options, and start A2L file generation by clicking the Generate A2L file button.

If you want to adapt an existing A2L file, you must also specify a Source A2L File. In this case, the text of the button changes to Adapt A2L file.


The following illustration shows the A2L File page as an example:



Tip

The RTI Bypass Blockset lets you import multiple A2L files at the same time. This allows you to import your normal ECU A2L file with all the ECU variable information, and an additional A2L file generated by the DCI Configuration Tool containing only the DCI-GSI2-specific interface information. This way, your original A2L file does not need to be modified at all.

It is also possible to automatically adapt the A2L file whenever a change in the file or in the DCI-GSI2 configuration is detected. To enable this feature, select the **Automatically update A2L file when necessary** checkbox.

In the most common cases, the options do not need to be modified. However, A2L files are complex and can already contain various IF_DATA and A2ML structures, which makes the fully automatic adaptation process complex as well. The options allow you to influence the adaptation process if you encounter any problems with the default options. For more details on the various options, refer to the [DCI Configuration](#)  document.

Tip

The A2L file adaptation process can also be initiated via the command line parameters of the DCI Configuration Tool. This allows you to fully automate the adaptation of the A2L file within your ECU software generation process.

A2L file structures concerned

The following information is added to the A2L file during generation or adaptation:

- A2ML description of the *XCP/XCPplus* and *dSPACE_XCP* IF_DATA structures
- The standardized *XCP/XCPplus* IF_DATA structure containing the XCP interface description
- The memory segments of *DATA* type are extended by XCP structures to allow data calibration via XCP
- An additional *dSPACE_XCP* IF_DATA structure (describes the additional dSPACE bypass features supported by the ECU)
- Description of the DCI-GSI2 system variables that contain status information on the DCI-GSI2

Related topics

HowTos

[How to Adapt an A2L File Automatically \(DCI Configuration\)](#) 

Bypass Model Configuration

Introduction

To perform bypassing with the DCI-GSI2, the *RTI Bypass Blockset* is used within MATLAB/Simulink to create the bypass model.

Configuring the Bypass Model

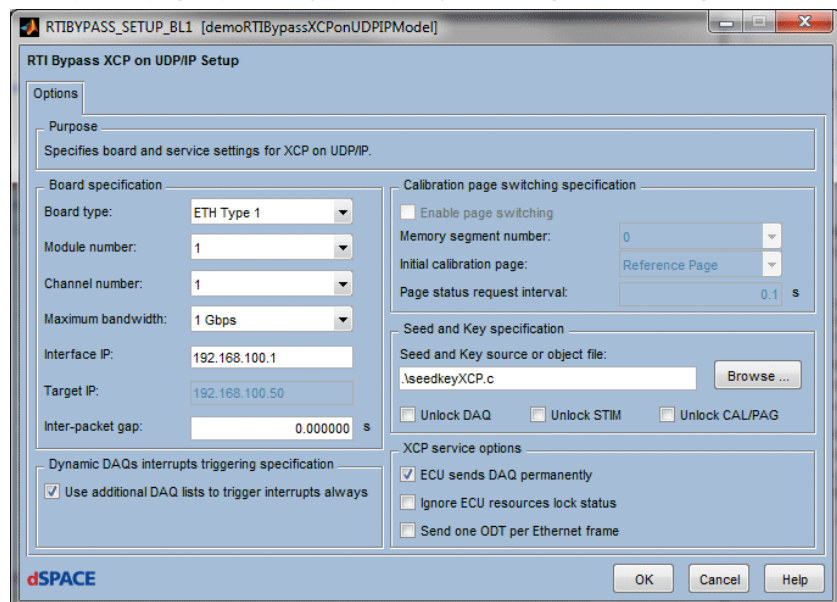
Introduction

After you have created the bypass model, you can build the bypass application which is then downloaded to the dSPACE rapid control prototyping hardware to perform the real-time function bypass.

Interface configuration in the RTI Bypass Setup block

After you have imported the A2L file as the database in the SETUP block of the RTI Bypass Blockset, select *XCP_on_UDP_IP* as the bypass interface. Then click Configure to specify bypass interface settings.

The Options page opens for you to specify the configuration settings:




- First, select the appropriate Board type depending on your RCP hardware. If you work with MicroAutoBox II and want to use its onboard Ethernet interface, select the *ETH Type 1* type. If you use the LVDS_CAB13/14 LVDS-to-Ethernet converter, select the *ECU Type 1 ETH* type and also the correct LVDS module.
- Set Maximum bandwidth to 1 Gbps.
- Specify a unique IP address for the Ethernet interface of the RCP system.

- Clear the Use additional DAQ lists to trigger interrupts always checkbox. It is not required with the DCI-GSI2 and would cause a minimal increase in bypass latencies.

All the other options can be left in their default state, which should be the same as shown in the screenshot above. The DCI-GSI2 does not have any Seed & Key protection.

Relevant blocks of the RTI Bypass Blockset

Apart from the SETUP and INFO blocks, only the INTERRUPT, READ and WRITE blocks are used for bypassing with the DCI-GSI2.

For more details on the blocks and information on modeling with the RTI Bypass Blockset, refer to the [RTI Bypass Blockset Reference](#) .

Note

Do not use the Upload and Download blocks for bypassing with the DCI-GSI2. They do not use the DAQ and STIM features of the XCP protocol. They would also not offer data consistency via the dSPACE Calibration and Bypassing Service, and would ultimately increase the bypass turnaround time.

Related topics

Basics

[Workflow for Setting Up a Service-Based Bypass.....](#) 26

Target Processor-Specific Configuration Settings

Introduction Provides details on some parameters of selected ECU target processors which are relevant to the DCI-GSI2 configuration.

Where to go from here

Information in this section

Cypress MB9D560 Family.....	60
Freescall/NXP MPC55xx/MPC56xx Family and STMicroelectronics SPC55xx/SPC56xx Family.....	62
Freescall/NXP MPC57xx Family and STMicroelectronics SPC57xx/SPC58xx Family.....	67
Infineon TriCore Family.....	71
Infineon XC2000 Family.....	76
Renesas M32R Family.....	78
Renesas RH850 Family.....	81
Renesas SH2 Family.....	85
Renesas SH2A/SH4A Family.....	88
Renesas V850E/V850E1 Family.....	91
Renesas V850E2 Family.....	93
Texas Instruments TMS570 Family.....	96
Toshiba TMPV770 Family.....	98

Cypress MB9D560 Family

Where to go from here

Information in this section

Basics on the Cypress MB9D560 Family.....	60
Provides basic information on the Cypress MB9D560 family.	
Generic Configuration Settings for the Cypress MB9D560 Target Processor Family.....	60
Provides information on configuration settings that are generic for all the processors of the Cypress MB9D560 family.	

Basics on the Cypress MB9D560 Family

Cypress MB9D560 family

The JTAG/CoreSight interface of this ARM microcontrollers is supported by the DCI-GSI2.

Related topics

References

Generic Configuration Settings for the Cypress MB9D560 Target Processor Family.....	60
---	--------------------

Generic Configuration Settings for the Cypress MB9D560 Target Processor Family

Generic configuration settings for the Cypress MB9D560 processor family

The following table lists configuration settings that are generic for all processors of the Cypress MB9D560 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	JTAG/ARM
Maximum JTAG frequency	10.0 MHz
Supported physical event channel types	<ul style="list-style-type: none">▪ Timer▪ ECU pin

Parameter	Value
Supported calibration method(s)	External RAM
Brain-dead flashing	Not supported

Related topics

Basics

Basics on the Cypress MB9D560 Family.....	60
---	----

Freescal/NXP MPC55xx/MPC56xx Family and STMicroelectronics SPC55xx/SPC56xx Family

Where to go from here

Information in this section

Basics on the Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families.....	62
Provides basic information on the MPC55xx and MPC56xx processor families from Freescal/NXP and the SPC55xx and SPC56xx processor families from STMicroelectronics.	
Generic Configuration Settings for the Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families.....	63
Provides information on configuration settings that are generic for all the processors of the Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx target processor families.	
Processor-Specific Configuration Settings (Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Families).....	64
Provides information on the configuration settings that are specific to the different target processor types of the Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx processor families.	

Basics on the Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families

Freescal/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx families

The MPC55xx and MPC56xx processor families from Freescal/NXP and the SPC55xx and SPC56xx processor families from STMicroelectronics come with a JTAG/Nexus debug interface. This interface allows real-time access to the ECU memory while the processor is running. It also provides a data trace interface that allows the DCI-GSI2 to record every ECU write access. This data tracing mechanism allows consistent data measurement without having to integrate a service into the ECU application.

There is an arbitration mechanism for the JTAG/Nexus interface that allows other tools, like a debugger, to be connected to the DCI-GSI2. The other tool has to support the arbitration mechanism as well.

Related topics**References**

Generic Configuration Settings for the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families.....	63
Processor-Specific Configuration Settings (Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Families).....	64

Generic Configuration Settings for the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families

Generic configuration settings for the MPC55xx/MPC56xx and SPC55xx/SPC56xx processor families

The following table lists configuration settings that are generic for all the processors of the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx target processor families. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Big endian (Motorola format)
Maximum JTAG frequency	40.0 MHz ¹⁾
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Watchpoint pin ▪ Watchpoint message ▪ Data trace write ▪ Data trace write value ▪ Fast variable overwrite
Supported calibration method(s)	External RAM
Supported overlay elements	Processor does not provide overlay elements.
Brain-dead flashing	Supported

¹⁾ The maximum JTAG frequency also depends on the target microcontroller clock, and might therefore be lower than the stated value.

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx target processor families, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Families\)](#) on page 64.

Related topics**Basics**

Basics on the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families..... 62

Processor-Specific Configuration Settings (Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Families)

Processor-specific configuration settings

The following table lists target-specific configuration settings for processors of the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx processor families:

Target Processor	READY Line	CENSOR Word Feature	MDO Pin Count	Available Trace Units
MPC5514 MPC5515 MPC5516 MPC5517	✓	–	4 / 8	<ul style="list-style-type: none"> ▪ CPU0 ▪ CPU1
MPC5534	✓	–	4 / 12	CPU0
MPC5554 MPC5553	✓	–	4 / 12	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0
MPC5561 MPC5565 MPC5566 MPC5567	✓	–	4 / 12	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0
MPC5602B MPC5602C MPC5603B MPC5603C MPC5604B MPC5604C SPC560B	–	–	2 / 4	CPU0
MPC5602D SPC560D	–	–	–	–
MPC5605B MPC5606B MPC5607B	–	–	4	CPU0
MPC5604E	–	✓	4	–
MPC5604P SPC560P	–	–	2 / 4	CPU0

Target Processor	READY Line	CENSOR Word Feature	MDO Pin Count	Available Trace Units
MPC5602S MPC5604S MPC5606S	–	–	4	CPU0
MPC5632M MPC5633M MPC5634M SPC563M	–	✓	4 / 12	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0
MPC5643L SPC564L	✓	✓	4 / 12	<ul style="list-style-type: none"> ▪ CPU0 ▪ CPU1
MPC5643A MPC5644A SPC564A	✓	✓	4 / 12	CPU0
MPC5644B MPC5644C MPC5645B MPC5645C MPC5646B MPC5646C SPC564B	✓	✓	8 / 12	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0 ▪ CPU1
MPC5668E MPC5668G	–	–	12	<ul style="list-style-type: none"> ▪ CPU0 ▪ CPU1
MPC5674F	✓	✓	12 / 16	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0 ▪ DMA1
MPC5671K MPC5673K MPC5675K	–	✓	4 / 12	<ul style="list-style-type: none"> ▪ CPU0 ▪ CPU1
MPC5676R	✓	✓	12 / 16	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0 ▪ CPU1 ▪ DMA1

For more information on the configuration parameters listed above, refer to [Target Interface Page \(DCI Configuration !\[\]\(919a2cb85b99741a73c0c31a427236a8_img.jpg\)](#)).

Generic configuration settings for the MPC55xx/MPC56xx and SPC55xx/SPC56xx processor families

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the MPC55xx/MPC56xx and SPC55xx/SPC56xx target processor families. Refer to [Generic Configuration Settings for the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics SPC55xx/SPC56xx Target Processor Families](#) on page 63.

Related topics

Basics

Basics on the Freescale/NXP MPC55xx/MPC56xx and STMicroelectronics
SPC55xx/SPC56xx Target Processor Families..... 62

Freescale/NXP MPC57xx Family and STMicroelectronics SPC57xx/SPC58xx Family

Where to go from here

Information in this section

[Basics on the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families..... 67](#)

Provides basic information on the MPC57xx processor family from Freescale/NXP and the SPC57xx and SPC58xx processor families from STMicroelectronics.

[Generic Configuration Settings for the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families..... 68](#)

Provides information on configuration settings that are generic for all the processors of the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx target processor families.

[Processor-Specific Configuration Settings \(Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Families\)..... 69](#)

Provides information on the configuration settings that are specific to the different target processor types of the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx processor families.

Basics on the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families

Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx families

The MPC57xx processor family from Freescale/NXP and the SPC57xx and SPC58xx processor families from STMicroelectronics come with a JTAG/Nexus debug interface. This interface allows real-time access to the ECU memory while the processor is running. It also provides a data trace interface that allows the DCI-GSI2 to record every ECU write access. This data tracing mechanism allows consistent data measurement without having to integrate a service into the ECU application.

There is an arbitration mechanism for the JTAG/Nexus interface that allows other tools, like a debugger, to be connected to the DCI-GSI2. The other tool has to support the arbitration mechanism as well.

Related topics

References

Generic Configuration Settings for the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families.....	68
Processor-Specific Configuration Settings (Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Families).....	69

Generic Configuration Settings for the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families

Generic configuration settings for the MPC57xx and SPC57xx/SPC58xx processor families

The following table lists configuration settings that are generic for all the processors of the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx target processor families. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Big endian (Motorola format)
Maximum JTAG frequency	40.0 MHz ¹⁾
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Poll event ▪ Watchpoint pin ▪ Watchpoint message ▪ Data trace write ▪ Data trace write value ▪ Fast variable overwrite
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Memory segments	The memory segments must be in the range 0x08A00000 - 0x09FFFFFF.
Overlay handle count	32
Overlay handle sizes	32 byte - 8 MB
Overlay memories	<ul style="list-style-type: none"> ▪ System RAM (at 0x40000000) ▪ Overlay RAM (at 0x0D000000) ▪ External RAM (at 0x0C000000)
Brain-dead flashing	Supported

¹⁾ The maximum JTAG frequency also depends on the target microcontroller clock, and might therefore be lower than the stated value.

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx target

processor families, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Freescal e/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Families\)](#) on page 69.

Related topics

Basics

Basics on the Freescal e/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families..... 67

Processor-Specific Configuration Settings (Freescal e/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Families)

Processor-specific configuration settings

The following table lists target-specific configuration settings for processors of the Freescal e/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx processor families:

Target Processor	READY Line	CENSOR Word Feature	MDO Pin Count	Available Trace Units
MPC5744P	✓	✓	4	CPU0
MPC5746M	–	✓	0	–
SPC57EM				
MPC5744K				
SPC574K				
MPC5746R				
MPC5748G				
MPC5775K				
MPC5777M				
MPC5777C				
SPC58xE	–	✓	0	–
SPC58xN				
SPC58xB				
SPC58xC				
SPC58xG				
SPC58xH				

For more information on the configuration parameters listed above, refer to [Target Interface Page \(DCI Configuration !\[\]\(8bba887393ca45b761e5cb49e755e762_img.jpg\)](#)).

Generic configuration settings for the MPC57xx and SPC57xx/SPC58xx processor families

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the MPC57xx and SPC57xx/SPC58xx target processor families. Refer to [Generic Configuration](#)

Settings for the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families on page 68.

Related topics

Basics

Basics on the Freescale/NXP MPC57xx and STMicroelectronics SPC57xx/SPC58xx Target Processor Families..... 67

Infineon TriCore Family

Where to go from here

Information in this section

Basics on the Infineon TriCore Family.....	71
Provides basic information on the Infineon TriCore family.	
Generic Configuration Settings for the Infineon TriCore Target Processor Family.....	71
Provides information on configuration settings that are generic for all the processors of the Infineon TriCore family.	
Processor-Specific Configuration Settings (Infineon TriCore Family).....	73
Provides information on the configuration settings that are specific to the different target processor types of the Infineon TriCore family.	
Additional Processor-Specific Information (Infineon TriCore Family).....	75
There are some points to note concerning some processors of the Infineon TriCore family.	

Basics on the Infineon TriCore Family

Infineon TriCore family

The TriCore family from Infineon comes with a JTAG/OCDS interface. Newer microcontrollers also have a Device Access Port (DAP) interface, which is faster and offers protection from transmission errors by means of a checksum.

Related topics

References

Additional Processor-Specific Information (Infineon TriCore Family).....	75
Generic Configuration Settings for the Infineon TriCore Target Processor Family.....	71
Processor-Specific Configuration Settings (Infineon TriCore Family).....	73

Generic Configuration Settings for the Infineon TriCore Target Processor Family

Generic configuration settings for the Infineon TriCore processor family

The following table lists configuration settings that are generic for all processors of the Infineon TriCore target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	<ul style="list-style-type: none"> ▪ JTAG/OCDS ▪ Device Access Port (DAP/DAP2)
Maximum interface frequency	<ul style="list-style-type: none"> ▪ 40.0 MHz for JTAG/OCDS ▪ 80.0 MHz for DAP ▪ 120.0 MHz for DAP2¹⁾
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Poll event ▪ Watchpoint pin ▪ Watchpoint polling²⁾
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Memory segments	The memory segments must be in the range 0x80000000 - 0x8FFFFFFF or 0xA0000000 - 0xAFFFFFFF.
Overlay handle count	<ul style="list-style-type: none"> ▪ 16 (for all microcontroller families except for the AURIX family) ▪ 32 (for the AURIX microcontroller family)
Brain-dead flashing	Supported

¹⁾ The DAP2 interface supports frequencies up to 160 MHz, but the DCI-GSI2 currently supports only frequencies up to 120 MHz.

²⁾ Supported for the AURIX microcontroller family only.

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Infineon TriCore target processor family, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Infineon TriCore Family\)](#) on page 73.

Related topics

Basics

[Basics on the Infineon TriCore Family](#)..... 71

Processor-Specific Configuration Settings (Infineon TriCore Family)

Production device microcontrollers configuration settings

The following table lists target-specific configuration settings for production device microcontrollers of the Infineon TriCore processor family:

Target Processor	BRKOUT Pins	Overlay Handle Sizes	Overlay RAM Address	Overlay RAM Size	DAP modes
TC1766	default	1 byte – 512 byte	0xC0000000	8 KB (0x2000)	–
TC1796	default	1 byte – 512 byte	0xC0000000	64 KB (0x10000)	–
TC1337 TC1367 TC1387 TC1724 TC1728 TC1736 TC1767 TC1782 TC1783 TC1784	P1.0 / P1.15	16 byte – 2 KB	0x8FE80000	8 KB (0x2000)	<ul style="list-style-type: none"> ▪ 2 pin mode ▪ 3 pin mode
TC1797	P9.13 / P9.14	16 byte – 2 KB	0x8FE80000	8 KB (0x2000)	<ul style="list-style-type: none"> ▪ 2 pin mode ▪ 3 pin mode
TC1791 TC1793 TC1798	P9.13 / P9.14	32 byte – 128 KB	0x90000000	128 KB (0x20000)	<ul style="list-style-type: none"> ▪ 2 pin mode ▪ 3 pin mode
TC21xx TC22xx TC23xx TC24xx TC26xx TC27xx TC29xx TC39xx TC38xx TC37xx TC36xx TC35xx TC33xx TC3Axx TC3Exx	TGO0 – TGO7	32 byte – 128 KB	0x90000000 (LMU) 0x70000000 (Core0) 0x60000000 (Core1) 0x50000000 (Core2)	The whole memory can be used as overlay memory. The available amount depends on the target type.	<ul style="list-style-type: none"> ▪ 2 pin mode ▪ 3 pin mode ▪ wide mode

For more information on the above configuration parameters, refer to [Target Interface Page \(DCI Configuration !\[\]\(d66ff64371a51729ac8c1cdaa685ba6f_img.jpg\)](#)).

For more information on the calibration parameters, refer to [Overlay RAM Calibration Page \(DCI Configuration !\[\]\(e3f8612927870f2e0f9f5989e6dd3064_img.jpg\)](#)).

**Emulation device
configuration settings**

TriCore microcontrollers with appended "ED" or "E" in the name (for example, TC1797ED or TC24XXE) are so-called emulation devices. They consist of the normal product chip part, and an *Emulation Extension Chip* (EEC) part. Such an emulation device has the same features as the product chip plus some special features offered by the EEC. One of those features is a big overlay RAM which can be used for data calibration by the DCI-GSI2.

In addition to the configuration settings and calibration parameters in the above table, the ED devices provide extra overlay RAM. If an external bus unit (EBU) is present, some TriCore microcontrollers can use external RAM as overlay memory as well. The following table shows the according calibration parameters:

Target Processor	ED Device Overlay Handle Sizes	Emulation Memory Address	Emulation Memory Size	External Memory Address
TC1766ED	1 KB – 32 KB	0x8FF20000	0x40000 (256 KB)	–
TC1796ED	1 KB – 32 KB	0x8FF00000	0x80000 (512 KB)	–
TC1337ED TC1367ED TC1387ED	1 KB – 128 KB	0x8FF00000	0x80000 (512 KB)	–
TC1724ED TC1728ED	1 KB – 128 KB	0x8FF00000	0x60000 (384 KB)	–
TC1736ED	1 KB – 128 KB	0x8FF00000	0x40000 (256 KB)	–
TC1767ED	1 KB – 128 KB	0x8FF00000	0x40000 (256 KB)	0x80800000
TC1782ED TC1783ED TC1784ED	1 KB – 128 KB	0x8FF00000	0x80000 (512 KB)	0x80800000
TC1797ED	1 KB – 128 KB	0x8FF00000	0x80000 (512 KB)	0x80800000
TC1791ED TC1793ED TC1798ED	32 byte – 128 KB	0x9F000000	0xC0000 (768 KB)	0x83000000
TC21xxE TC22xxE TC23xxE TC24xxE	32 byte – 128 KB	0x9F000000	0x40000 (256 KB)	–
TC26xxE	32 byte – 128 KB	0x9F000000	0x80000 (512 KB)	–
TC27xxE	32 byte – 128 KB	0x9F000000	0x100000 (1 MB)	–
TC29xxE	32 byte – 128 KB	0x9F000000	0x200000 (2 MB)	–
TC39xxE	32 byte – 128 KB	0x99000000	0x400000 (4 MB)	0x82000000
TC37xxE	32 byte – 128 KB	0x99000000	0x300000 (3 MB)	0x82000000
TC33xxE	32 byte – 128 KB	0x99000000	0x100000 (1 MB)	0x82000000

Generic configuration settings for the Infineon TriCore processor family

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the Infineon TriCore target processor family. Refer to [Generic Configuration Settings for the Infineon TriCore Target Processor Family](#) on page 71.

Related topics

Basics

[Basics on the Infineon TriCore Family](#)..... 71

References

[Additional Processor-Specific Information \(Infineon TriCore Family\)](#)..... 75

Additional Processor-Specific Information (Infineon TriCore Family)

Additional processor-specific information on Infineon TriCore microcontrollers

The following notes apply to some processors of the Infineon TriCore family:

Note

Early versions of the TC1791, TC1793 and TC1798 microcontrollers have a bug that prevents the DCI-GSI2 from accessing the overlay control registers via the serial interface. As a workaround, dSPACE Calibration and Bypassing Service 2.2 or later can be used to perform the necessary read/write accesses to the registers.

When the command processor part of the *dSPACE Calibration and Bypassing Service* is integrated into the ECU application, make the following setting in the `dsECUcfg.h` file:

- Set `DSECU_CALIBRATION_COMMANDS` to `'DSECU_ENABLED'`.
- Set `DSECU_CAL_MEMORY_COPY_METHOD` to `'DSECU_MEMCPY_REGISTER'`.

For further information, refer to the [dSPACE Calibration and Bypassing Service Implementation](#)  document.

Related topics

Basics

[Basics on the Infineon TriCore Family](#)..... 71

References

[Processor-Specific Configuration Settings \(Infineon TriCore Family\)](#)..... 73

Infineon XC2000 Family

Where to go from here

Information in this section

- [Basics on the Infineon XC2000 Family..... 76](#)
Provides basic information on the Infineon XC2000 family.
- [Generic Configuration Settings for the Infineon XC2000 Target Processor Family..... 76](#)
Provides information on configuration settings that are generic for all the processors of the Infineon XC2000 family.

Basics on the Infineon XC2000 Family

Infineon XC2000 family

The Infineon XC2000 family is a 16-bit microcontroller family that supports the robust Device Access Port (DAP) interface.

Related topics

References

- [Generic Configuration Settings for the Infineon XC2000 Target Processor Family..... 76](#)

Generic Configuration Settings for the Infineon XC2000 Target Processor Family

Generic configuration settings for the Infineon XC2000 processor family

The following table lists configuration settings that are generic for all processors of the Infineon XC2000 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	Device Access Port (DAP)
Maximum interface frequency	80.0 MHz
Supported DAP modes	2 pin mode

Parameter	Value
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Watchpoint pin
BRKOUT Pins	<ul style="list-style-type: none"> ▪ P1.5 ▪ P6.0 ▪ P9.3 ▪ P10.11
Supported calibration method(s)	External RAM
Brain-dead flashing	Supported

Related topics

Basics

[Basics on the Infineon XC2000 Family..... 76](#)

Renesas M32R Family

Where to go from here

Information in this section

Basics on the Renesas M32R Family.....	78
Provides basic information on the Renesas M32R family.	
Generic Configuration Settings for the Renesas M32R Target Processor Family.....	78
Provides information on configuration settings that are generic for all the processors of the Renesas M32R family.	
Processor-Specific Configuration Settings (Renesas M32R Family).....	79
Provides information on the configuration settings that are specific to the different target processor types of the Renesas M32R family.	

Basics on the Renesas M32R Family

Renesas M32R family

Renesas M32R microcontrollers with an NBD interface are supported by the DCI-GSI2.

Related topics

Basics

Processor-Specific Configuration Settings (Renesas M32R Family).....	79
--	--------------------

References

Generic Configuration Settings for the Renesas M32R Target Processor Family.....	78
--	--------------------

Generic Configuration Settings for the Renesas M32R Target Processor Family

Generic configuration settings for the Renesas M32R processor family

The following table lists configuration settings that are generic for all processors of the Renesas M32R target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Big endian (Motorola format)
Serial interface	NBD

Parameter	Value
Maximum interface frequency	12.5 MHz
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Watchpoint pin
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Brain-dead flashing	Not supported

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Renesas M32R target processor family, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Renesas M32R Family\)](#) on page 79.

Related topics

Basics

[Basics on the Renesas M32R Family..... 78](#)

Processor-Specific Configuration Settings (Renesas M32R Family)

Processor-specific configuration settings

The following table lists target-specific configuration settings for microcontrollers of the Renesas M32R family:

Target Processor	Overlay Handle Count	Overlay RAM Address
M32185F4	4	0x804000
M32186F8	8	0x804000
M32192F8	16	0x810000
M32196F8	8	0x804000

Generic configuration settings for the Renesas M32R processor family

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the M32R target processor family. Refer to [Generic Configuration Settings for the Renesas M32R Target Processor Family](#) on page 78.

Related topics

Basics

Basics on the Renesas M32R Family..... 78

Renesas RH850 Family

Where to go from here	Information in this section
	Basics on the Renesas RH850 Family..... 81 Provides basic information on the Renesas RH850 family.
	Generic Configuration Settings for the Renesas RH850 Target Processor Family..... 81 Provides information on configuration settings that are generic for all the processors of the Renesas RH850 family.
	Processor-Specific Configuration Settings (Renesas RH850 Family)..... 82 Provides information on the configuration settings that are specific to the different target processor types of the Renesas RH850 family.

Basics on the Renesas RH850 Family

Renesas RH850 family	Renesas RH850 microcontrollers with a JTAG/Nexus interface or NBD interface are supported by the DCI-GSI2.
----------------------	--

Related topics	Basics
	Processor-Specific Configuration Settings (Renesas RH850 Family)..... 82
	References
	Generic Configuration Settings for the Renesas RH850 Target Processor Family..... 81

Generic Configuration Settings for the Renesas RH850 Target Processor Family

Generic configuration settings for the Renesas RH850 processor family	The following table lists configuration settings that are generic for all processors of the Renesas RH850 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.
---	---

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	<ul style="list-style-type: none"> ▪ JTAG/Nexus ▪ NBD
Maximum interface frequency	<ul style="list-style-type: none"> ▪ 40.0 MHz for RH850/E2x processors ▪ 25.0 MHz for all other processors
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Watchpoint pin ▪ Polling event
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Brain-dead flashing	<ul style="list-style-type: none"> ▪ Supported for JTAG/Nexus ▪ Not supported for NBD

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Renesas RH850 target processor family, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Renesas RH850 Family\)](#) on page 82.

Related topics

Basics

[Basics on the Renesas RH850 Family..... 81](#)

Processor-Specific Configuration Settings (Renesas RH850 Family)

Processor-specific configuration settings

The following table lists target-specific configuration settings for microcontrollers of the Renesas RH850 family:

Target Processor	Overlay Handle Sizes	Overlay RAM Address	Overlay RAM Size
RH850/E1x-FCC1-EVA RH850/E1MS-FCC1 RH850/E1L24-FCC1 RH850/E1L16-FCC1	<ul style="list-style-type: none"> ▪ 8 KB ▪ 32 KB 	0xF9800000	768 KB
RH850/E1M-S RH850/E1M-S2 RH850/E1L24 RH850/E1L16	8 KB	0xF9800000	8 KB

Target Processor	Overlay Handle Sizes	Overlay RAM Address	Overlay RAM Size
RH850/E1x-FCC2-EVA RH850/E1L-FCC2 RH850/E1M-FCC2 RH850/E1M-A-FCC2	32 KB	0xF9800000	1024 KB
RH850/E1M-S2 RH850/E1M-A	8 KB	0xF9800000	8 KB
RH850/E2x-FCC1-EVA RH850/E2L-FCC1 RH850/E2M-FCC1 RH850/E2X-FCC2-EVA RH850/E2H-FCC2 RH850/E2UH-FCC2	<ul style="list-style-type: none"> ▪ 8 KB ▪ 32 KB 	0xFB000000	1024 KB
RH850/E2L RH850/E2M RH850/E2H RH850/E2UH	–	–	–
RH850/F1H RH850/F1K RH850/F1L RH850/F1L-G RH850/F1M RH850/F1KM RH850/F1KH	–	–	–
RH850/P1M RH850/P1M-E	8 KB	0xF9800000	8 KB ¹⁾
RH850/P1M-C RH850/P1H-C	–	–	–
RH850/P1M-CE RH850/P1H-CE	32 KB	0xF9800000	2048 KB
RH850/C1M RH850/C1M-A1 RH850/C1M-A2 RH850/C1M-AZ RH850/C1H	–	–	–

¹⁾ There are variants of this microcontroller with 32 KB of overlay RAM available.

Generic configuration settings for the Renesas RH850 processor family

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the RH850 target processor family. Refer to [Generic Configuration Settings for the Renesas RH850 Target Processor Family](#) on page 81.

Related topics

Basics

[Basics on the Renesas RH850 Family..... 81](#)

Renesas SH2 Family

Where to go from here

Information in this section

Basics on the Renesas SH2 Family.....	85
Provides basic information on the Renesas SH2 family.	
Generic Configuration Settings for the Renesas SH2 Target Processor Family.....	85
Provides information on configuration settings that are generic for all the processors of the Renesas SH2 family.	
Processor-Specific Configuration Settings (Renesas SH2 Family).....	86
Provides information on the configuration settings that are specific to the different target processor types of the Renesas SH2 family.	

Basics on the Renesas SH2 Family

Renesas SH2 family

Renesas SH2 microcontrollers with an NBD interface are supported by the DCI-GSI2.

Related topics

Basics

Processor-Specific Configuration Settings (Renesas SH2 Family).....	86
---	--------------------

References

Generic Configuration Settings for the Renesas SH2 Target Processor Family.....	85
---	--------------------

Generic Configuration Settings for the Renesas SH2 Target Processor Family

Generic configuration settings for the Renesas SH2 processor family

The following table lists configuration settings that are generic for all processors of the Renesas SH2 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Big endian (Motorola format)
Serial interface	NBD

Parameter	Value
Maximum interface frequency	10.0 MHz
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Brain-dead flashing	Not supported

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Renesas SH2 target processor family, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Renesas SH2 Family\)](#) on page 86.

Related topics

Basics

[Basics on the Renesas SH2 Family..... 85](#)

Processor-Specific Configuration Settings (Renesas SH2 Family)

Processor-specific configuration settings

The following table lists target-specific configuration settings for microcontrollers of the Renesas SH2 family:

Target Processor	Overlay Handle Count	Overlay Handle Sizes	Overlay RAM Address
SH7047 SH7049	1	0x1000	0xFFFFD000
SH7055	1	0x1000	0xFFFF6000
SH7058 SH7058S	1	0x1000	0xFFFF0000
SH7058FCC	8	0x1000	0xFFFE8000
SH7058RFCC	16	0x8000	0xFFFB0000
SH7059	1	0x8000	0xFFFE8000
SH7059FCC	24	0x8000	0xFFFB4000
SH7147	1	0x1000	0xFFFFA000

Generic configuration settings for the Renesas SH2 processor family

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the SH2 target processor family.

Refer to [Generic Configuration Settings for the Renesas SH2 Target Processor Family](#) on page 85.

Related topics

Basics

[Basics on the Renesas SH2 Family..... 85](#)

Renesas SH2A/SH4A Family

Where to go from here

Information in this section

Basics on the Renesas SH2A/SH4A Family.....	88
Provides basic information on the Renesas SH2A/SH4A family.	
Generic Configuration Settings for the Renesas SH2A/SH4A Target Processor Family.....	88
Provides information on configuration settings that are generic for all the processors of the Renesas SH2A/SH4A family.	
Processor-Specific Configuration Settings (Renesas SH2A/SH4A Family).....	89
Provides information on the configuration settings that are specific to the different target processor types of the Renesas SH2A/SH4A family.	

Basics on the Renesas SH2A/SH4A Family

Renesas SH2A/SH4A family

Renesas SH2A/SH4A microcontrollers with an NBD interface or JTAG/H-UDI interface are supported by the DCI-GSI2.

Related topics

Basics

Processor-Specific Configuration Settings (Renesas SH2A/SH4A Family).....	89
---	--------------------

References

Generic Configuration Settings for the Renesas SH2A/SH4A Target Processor Family.....	88
---	--------------------

Generic Configuration Settings for the Renesas SH2A/SH4A Target Processor Family

Generic configuration settings for the Renesas SH2A/SH4A processor family

The following table lists configuration settings that are generic for all processors of the Renesas SH2A/SH4A target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Big endian (Motorola format)
Serial interface	<ul style="list-style-type: none"> ▪ NBD ▪ JTAG/H-UDI
Maximum interface frequency	<ul style="list-style-type: none"> ▪ 25.0 MHz for NBD ▪ 20.0 MHz for JTAG/H-UDI
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Poll event (only for NBD)
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Brain-dead flashing	<ul style="list-style-type: none"> ▪ Not supported for NBD ▪ Supported for JTAG/H-UDI

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Renesas SH2A/SH4A target processor family, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Renesas SH2A/SH4A Family\)](#) on page 89.

Related topics

Basics

[Basics on the Renesas SH2A/SH4A Family..... 88](#)

Processor-Specific Configuration Settings (Renesas SH2A/SH4A Family)

Processor-specific configuration settings

The following table lists target-specific configuration settings for microcontrollers of the Renesas SH2A/SH4A family:

Target Processor	Overlay Count	Overlay Size	Overlay Address
SH72513	0	0	—
SH72513FCC SH72518 SH72523EVA	8	0x10000	0x00600000
SH72531	0	0	—
SH72531FCC SH72533FCC SH72543 SH72546RFCC SH72556REVA	8	0x10000	0x00600000

Target Processor	Overlay Count	Overlay Size	Overlay Address
SH72567RFCC			
SH72567R	0	0	—
SH74513 SH7462	0	0	—

**Generic configuration
settings for the Renesas
SH2A/SH4A processor family**

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the SH2A/SH4A target processor family. Refer to [Generic Configuration Settings for the Renesas SH2A/SH4A Target Processor Family](#) on page 88.

Related topics**Basics**

[Basics on the Renesas SH2A/SH4A Family..... 88](#)

Renesas V850E/V850E1 Family

Where to go from here

Information in this section

Basics on the Renesas V850E/V850E1 Family.....	91
Provides basic information on the Renesas V850E/V850E1 family.	
Generic Configuration Settings for the Renesas V850E/V850E1 Target Processor Family.....	91
Provides information on configuration settings that are generic for all the processors of the Renesas V850E/V850E1 family.	

Basics on the Renesas V850E/V850E1 Family

Renesas V850E/V850E1 family

Renesas V850E/V850E1 microcontrollers with an NBD interface are supported by the DCI-GSI2.

These microcontrollers support calibration via overlay units. It is not necessary to configure any overlay handles when using the overlay RAM calibration method. The DCI-GSI2 will automatically use the overlay units provided by the target microcontroller.

Related topics

References

Generic Configuration Settings for the Renesas V850E/V850E1 Target Processor Family.....	91
--	----

Generic Configuration Settings for the Renesas V850E/V850E1 Target Processor Family

Generic configuration settings for the Renesas V850E/V850E1 processor family

The following table lists configuration settings that are generic for all processors of the Renesas V850E/V850E1 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	NBD
Maximum interface frequency	20.0 MHz
Supported physical event channel types	<ul style="list-style-type: none">▪ Timer▪ ECU pin▪ Watchpoint pin
Supported calibration method(s)	<ul style="list-style-type: none">▪ Overlay RAM▪ External RAM
Brain-dead flashing	Not supported

Related topics

Basics

[Basics on the Renesas V850E/V850E1 Family..... 91](#)

Renesas V850E2 Family

Where to go from here	Information in this section
	Basics on the Renesas V850E2 Family..... 93
	Provides basic information on the Renesas V850E2 family.
	Generic Configuration Settings for the Renesas V850E2 Target Processor Family..... 93
	Provides information on configuration settings that are generic for all the processors of the Renesas V850E2 family.
	Processor-Specific Configuration Settings (Renesas V850E2 Family)..... 94
	Provides information on the configuration settings that are specific to the different target processor types of the Renesas V850E2 family.

Basics on the Renesas V850E2 Family

Renesas V850E2 family	Renesas V850E2 microcontrollers with a JTAG/Nexus interface are supported by the DCI-GSI2.
-----------------------	--

Related topics	Basics
	Processor-Specific Configuration Settings (Renesas V850E2 Family)..... 94
	References
	Generic Configuration Settings for the Renesas V850E2 Target Processor Family..... 93

Generic Configuration Settings for the Renesas V850E2 Target Processor Family

Generic configuration settings for the Renesas V850E2 processor family	The following table lists configuration settings that are generic for all processors of the Renesas V850E2 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.
--	--

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	JTAG/Nexus
Supported physical event channel types	<ul style="list-style-type: none"> ▪ Timer ▪ ECU pin ▪ Watchpoint pin ▪ Watchpoint message ▪ Data trace write ▪ Data trace write value ▪ Fast variable overwrite ▪ Polling event
Supported calibration method(s)	<ul style="list-style-type: none"> ▪ Overlay RAM ▪ External RAM
Brain-dead flashing	Supported
MDO pin count	2, 4, 8, 16, 24, or 48 ¹⁾
MSEO pin count	1, 2

¹⁾ 24 and 48 MDO pins are currently not supported by the DCI-GSI2.

Processor-specific configuration settings

In addition to the above generic configuration settings for processors of the Renesas V850E2 target processor family, there are parameters that are specific to the different target processor types. Refer to [Processor-Specific Configuration Settings \(Renesas V850E2 Family\)](#) on page 94.

Related topics

Basics

[Basics on the Renesas V850E2 Family..... 93](#)

Processor-Specific Configuration Settings (Renesas V850E2 Family)

Processor-specific configuration settings

Some configuration settings are target-specific to the type of microcontroller of the Renesas V850E2 family.

Target Processor	Available Trace Units	Maximum Interface Frequency
V850E2/Px4	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0 	25.0 MHz
V850E2/Px4-L	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0 	25.0 MHz
V850E2/Fx4	<ul style="list-style-type: none"> ▪ CPU0 ▪ DMA0 	20.0 MHz

Target Processor	Available Trace Units	Maximum Interface Frequency
V850E2/Fx4-H	<ul style="list-style-type: none"> CPU0 DMA0 	20.0 MHz
V850E2/Fx4-L	<ul style="list-style-type: none"> CPU0 DMA0 	20.0 MHz
V850E2/Fx4-G	<ul style="list-style-type: none"> CPU0 DMA0 	20.0 MHz
V850E2/Mx4	<ul style="list-style-type: none"> CPU0 DMA0 	25.0 MHz
V850E2/CS2	<ul style="list-style-type: none"> CPU0 DMA0 	25.0 MHz

Tuning RAM To perform ECU calibration via overlay units of the ECU processor with the DCI-GSI2, the target microcontroller must have an overlay RAM. Renesas calls the overlay RAM *tuning RAM*. Some microcontrollers do not have any tuning RAM, some have only a small amount of it.

Below is an example configuration for the overlay handles of a V850E2 microcontroller:

Overlay Handle Number	Overlay Handle Address	Overlay Handle Size
0	0x1A800000	0x1000
1	0x1A801000	0x1000
2	0x1A802000	0x1000
...
31	0x1A81F000	0x1000

For details, refer to the hardware manual of your Renesas microcontroller.

Generic configuration settings for the Renesas V850E2 processor family

In addition to the above processor-specific configuration settings, there are parameters that are generic for all processors of the V850E2 target processor family. Refer to [Generic Configuration Settings for the Renesas V850E2 Target Processor Family](#) on page 93.

Related topics

Basics

[Basics on the Renesas V850E2 Family](#)..... 93

Texas Instruments TMS570 Family

Where to go from here

Information in this section

- [Basics on the Texas Instruments TMS570 Family..... 96](#)
Provides basic information on the Texas Instruments TMS570 family.
- [Generic Configuration Settings for the Texas Instruments TMS570 Target Processor Family..... 96](#)
Provides information on configuration settings that are generic for all the processors of the Texas Instruments TMS570 family.

Basics on the Texas Instruments TMS570 Family

Texas Instruments TMS570 family

The JTAG/CoreSight interface of this ARM microcontrollers is supported by the DCI-GSI2.

Related topics

References

- [Generic Configuration Settings for the Texas Instruments TMS570 Target Processor Family..... 96](#)

Generic Configuration Settings for the Texas Instruments TMS570 Target Processor Family

Generic configuration settings for the Texas Instruments TMS570 processor family

The following table lists configuration settings that are generic for all processors of the Texas Instruments TMS570 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Big endian (Motorola format)
Serial interface	JTAG/ARM
Maximum JTAG frequency	12.0 MHz
Supported physical event channel types	<ul style="list-style-type: none">▪ Timer▪ ECU pin

Parameter	Value
Supported calibration method(s)	External RAM
Brain-dead flashing	Not supported

Related topics

Basics

[Basics on the Texas Instruments TMS570 Family.....](#) 96

Toshiba TMPV770 Family

Where to go from here

Information in this section

Basics on the Toshiba TMPV770 Family.....	98
Provides basic information on the Toshiba TMPV770 family.	
Generic Configuration Settings for the Toshiba TMPV770 Target Processor Family.....	98
Provides information on configuration settings that are generic for all the processors of the Toshiba TMPV770 family.	

Basics on the Toshiba TMPV770 Family

Toshiba TMPV770 family

The JTAG/CoreSight interface of this ARM microcontrollers is supported by the DCI-GSI2.

Related topics

References

Generic Configuration Settings for the Toshiba TMPV770 Target Processor Family.....	98
---	--------------------

Generic Configuration Settings for the Toshiba TMPV770 Target Processor Family

Generic configuration settings for the Toshiba TMPV770 processor family

The following table lists configuration settings that are generic for all processors of the Toshiba TMPV770 target processor family. Only settings that are relevant for the DCI-GSI2 configuration are listed.

Parameter	Value
Endianness	Little endian (Intel format)
Serial interface	JTAG/ARM
Maximum JTAG frequency	30.0 MHz
Supported physical event channel types	<ul style="list-style-type: none">▪ Timer▪ ECU pin

Parameter	Value
Supported calibration method(s)	External RAM
Brain-dead flashing	Not supported

Related topics

Basics

Basics on the Toshiba TMPV770 Family.....	98
---	----

Preparing a Support Request

Introduction

If you have difficulties with your DCI-GSI2 which you cannot solve on your own, contact dSPACE Support by sending a support request.

Creating a Support Report

Introduction

If you have difficulties with your device which you cannot solve on your own, contact dSPACE Support by sending a support request. Aside from a description of the problem and information on how to reproduce it, your e-mail should also contain a "support report" generated by the DCI Configuration Tool.

Note

Alternatively, you can create support information for the connected DCI-GSI2 device using the dSPACE Installation Manager's diagnostic feature.

Support report

To help dSPACE Support in analyzing the observed problem and to speed up your support request, you should attach a "support report" generated by the DCI Configuration Tool to your e-mail.

The support report contains textual information on the configuration parameters of the device used, version information on dSPACE software components, and device-internal status information. For detailed information on how to generate a support report via the DCI Configuration Tool, refer to [How to Create a Support Report \(DCI Configuration !\[\]\(47734e4656765d20df4fdbd5b7aff048_img.jpg\)](#)).

Related topics

HowTos

[How to Create a Support Report \(DCI Configuration !\[\]\(eafc244b53721dd1ec133f0772f70fc7_img.jpg\)](#))

A

A2L file for DCI-GSI2
preparing 21

C

Common Program Data folder 8
configuring a DCI-GSI2 using the DCI
Configuration Tool 18
configuring the DCI-GSI2 19
creating a support report 101

D

DCI-GSI2
initial setup 16
optimizing the configuration settings 19
Documents folder 8

I

initial setup of DCI-GSI2 16

L

Local Program Data folder 8

O

optimizing the DCI-GSI2 configuration
settings 19

P

parameters of ECU target processors 59
preparing A2L file for DCI-GSI2 21

S

safety precautions 11
setting up a bypass with a DCI-GSI2 23
adapting or generating an A2L file 55
configuring the DCI-GSI2 47
creating the bypass model 57
integrating the service as binary code 41
integrating the service source code 28
manual service integration 28
service integration via ECU Interface
Manager 41
testing the integrated service 53
workflow 26
support report 101

