

DS4121 ECU Interface Board

# RTLib Reference

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:	<a href="mailto:info@dspace.de">info@dspace.de</a>
Web:	<a href="http://www.dspace.com">http://www.dspace.com</a>

## 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](mailto: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.

© 2000 - 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 Document	5
ECU Interface Unit	7
Basics.....	8
DS4121 RTLib Functions.....	8
Data Layout in the DPMEM.....	9
System Functions.....	11
ds4121_init.....	11
ds4121_connection_check.....	12
ds4121_fifo_check.....	13
ds4121_fifo_reset.....	14
ds4121_fifo_start.....	15
ds4121_ecu_version_check.....	16
Interrupt Handling.....	18
ds4121_subint_init.....	18
ds4121_end_of_int_set.....	19
Reading Data from the ECU.....	21
ds4121_p_int_read8.....	21
ds4121_s_int_read8.....	22
ds4121_le_p_int_read16, ds4121_le_p_int_read32, ds4121_le_s_int_read16, ds4121_le_s_int_read32.....	23
ds4121_be_p_int_read16, ds4121_be_p_int_read32, ds4121_be_s_int_read16, ds4121_be_s_int_read32.....	24
ds4121_le_p_fl_read32, ds4121_le_s_fl_read32, ds4121_be_p_fl_read32, ds4121_be_s_fl_read32.....	25
Transferring Data to the ECU.....	27
ds4121_p_int_write8.....	27
ds4121_s_int_write8.....	29
ds4121_le_p_int_write16, ds4121_le_p_int_write32, ds4121_le_s_int_write16, ds4121_le_s_int_write32.....	30
ds4121_be_p_int_write16, ds4121_be_p_int_write32, ds4121_be_s_int_write16, ds4121_be_s_int_write32.....	31
ds4121_le_p_fl_write32, ds4121_le_s_fl_write32, ds4121_be_p_fl_write32, ds4121_be_s_fl_write32.....	32

## ECU Software Porting Kit 35

Generating the ECU Specific File.....	36
Defines to be Specified in the dsECUframe.c File.....	36
Initializing the ECU Software Porting Kit.....	37
DSECU_SINT_INIT.....	37
dsecu_define_sender.....	39
dsecu_startup.....	40
dsecu_alive.....	41
dsecu_sender_suspend.....	43
Subinterrupt Programming.....	45
DSECU_SINT_SEND.....	45
dsecu_sint_send.....	46
DSECU_EOSI_POLL.....	47
dsecu_eosi_poll.....	48
Host Utilities.....	50
dsECUnew.bat.....	50

## Word-Based Subinterrupt Handling 51

dssint_wb_define_receiver.....	52
dssint_wb_subint_disable.....	53
dssint_wb_subint_enable.....	54
dssint_wb_decode.....	55
dssint_wb_acknowledge.....	56
dssint_wb_subint_reset.....	57
dssint_wb_end_of_sint_set.....	58
dssint_wb_startup.....	58
dssint_wb_alive.....	60
dssint_wb_suspend.....	61

## Function Execution Times 63

Function Execution Times.....	63
-------------------------------	----

## Index 67









# About This Document

## Content

The DS4121 Real-Time Library (RTLib) provides the C functions and macros you need to program the DS4121 ECU Interface Board.

## Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
	Indicates a hazard that, if not avoided, could result in property damage.
	Indicates important information that you should take into account to avoid malfunctions.
	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](http://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.

# ECU Interface Unit

## Where to go from here

## Information in this section

<a href="#">Basics.....</a>	<a href="#">8</a>
To get information on the data types, the function names used, and the data layout of the DS4121's dual-port memory (DPMEM).	
<a href="#">System Functions.....</a>	<a href="#">11</a>
To get information on the initialization of the board, how to check the connection to the ECU and to set the mode of the onboard or external DPMEM.	
<a href="#">Interrupt Handling.....</a>	<a href="#">18</a>
To get information on the functions used for interrupt handling.	
<a href="#">Reading Data from the ECU.....</a>	<a href="#">21</a>
To get information on the data-typed functions for reading data from the DPMEM.	
<a href="#">Transferring Data to the ECU.....</a>	<a href="#">27</a>
To get information on the data-typed functions for writing data to the DPMEM.	

# Basics

## Where to go from here

## Information in this section

<a href="#">DS4121 RTLib Functions.....</a>	<a href="#">8</a>
To get an overview of the data types and the function names used.	
<a href="#">Data Layout in the DPMEM.....</a>	<a href="#">9</a>
Provides information on the data layout of the board's dual-port memory (DPMEM). The DPMEM provides read and write access to ECU variables for the ECU and the board.	

## DS4121 RTLib Functions

### Introduction

RTLib functions provide data-typed, format-dependent read and write access to the DPMEM.

### Abbreviations

The function names characterize the supported functionality with the following abbreviations:

Abbreviation	Meaning
ob	access to the <i>onboard</i> DPMEM
ext	access to an <i>external</i> DPMEM
p	<i>packed</i> data
s	<i>scattered</i> data
le	<i>little</i> endian
be	<i>big</i> endian
int	<i>integer</i>
fl	<i>floating</i> point
8, 16, 32	number of bits for a data word

### Example

The `ds4121_p_int_write8` function writes 8-bit *integer* data in a *packed* format to the DPMEM.

### Error messages

For some functions error messages are defined. These messages are reported by ControlDesk. Each message comprises the function name and the PHS-bus offset



(0xXX) of the DS4121 that generates the message. Furthermore, we will give a short description of the reason in this reference.

#### Example

**ds4121\_init(0xXX): Board not found!** The DS4121 was not found with the given PHS-bus offset.

#### Related topics

Basics

[Data Layout in the DPMEM.....9](#)

## Data Layout in the DPMEM

#### Introduction

The DS4121 has a 16-bit dual-port memory (DPMEM) which provides read and write access to ECU variables for the ECU and the DS4121. The data layout in the DPMEM, i.e., the way ECU variables are stored in, written to, and read from the DPMEM, depends on the ECU microprocessor's architecture.

#### ECU data format

Depending on the microcontroller's bus width, the DS4121 provides byte-wise and/or word-wise data access to the DPMEM. ECU data values are arranged in packed and/or scattered format in the DPMEM. If the ECU values' bits are scattered, only byte-wise (8-bit) data access to the DPMEM is possible. Packed format implies word-wise (16-bit) data access. Thus, an ECU value might need to be divided into 8-bit or 16-bit segments which are stored at two or more different DPMEM locations. To read an ECU data value from the DPMEM, it might be necessary to read bits from various DPMEM locations and put them together to form a single value. To write a single value to the DPMEM, it might be necessary to divide it into several 8- or 16-bit segments and write the bits byte- or word-wise to several DPMEM locations (see example below).

#### Data arrangement in the DPMEM

The arrangement of ECU values in the DPMEM depends on the data format and the data bus format used by the ECU. The following factors affect the memory addressing:

- Width of the ECU variable (8-bit, 16-bit or 32-bit data)
- Format of the ECU data (packed or scattered data)
- Byte order (little endian or big endian)

**Example**

This example shows the arrangement of 8-bit, 16-bit and 32-bit ECU data values in a 16-bit DPMEM according to different ECU data formats:

- Arrangement of 8-bit data values in the DPMEM

Value	Data Format	DPMEM
0x12	Packed, upper halfword	[0] 0x1200
	Packed, lower halfword	[0] 0x0012
	Scattered	[0] 0x0012

- Arrangement of 16-bit data values in the DPMEM

Value	Data Format	Byte Order	DPMEM
0x1234	Packed	Little endian	[0] 0x1234
	Packed	Big endian	[0] 0x1234
	Scattered	Little endian	[0] 0x0012
			[1] 0x0034
	Scattered	Big endian	[0] 0x0012
			[1] 0x0034

- Arrangement of 32-bit data values in the DPMEM

Value	Data Format	Byte Order	DPMEM
0x12345678	Packed	Little endian	[0] 0x5678
			[1] 0x1234
	Packed	Big endian	[0] 0x1234
			[1] 0x5678
	Scattered	Little endian	[0] 0x0056
			[1] 0x0078
			[2] 0x0012
			[3] 0x0034
	Scattered	Big endian	[0] 0x0012
			[1] 0x0034
			[2] 0x0056
			[3] 0x0078

**Related topics****Basics**

[DS4121 RTLib Functions..... 8](#)

# System Functions

## Introduction

System functions allow you to initialize the board, to check the connection to the ECU and to set the mode of the onboard or external DPMEM.

## Where to go from here

## Information in this section

<a href="#">ds4121_init.....</a>	<a href="#">11</a>
To initialize the DS4121 board.	
<a href="#">ds4121_connection_check.....</a>	<a href="#">12</a>
To check the connection to the ECU.	
<a href="#">ds4121_fifo_check.....</a>	<a href="#">13</a>
To check the FIFO state of a DS4121 channel.	
<a href="#">ds4121_fifo_reset.....</a>	<a href="#">14</a>
To reset the FIFO of a DS4121 channel.	
<a href="#">ds4121_fifo_start.....</a>	<a href="#">15</a>
To start the FIFO of a DS4121 channel.	
<a href="#">ds4121_ecu_version_check.....</a>	<a href="#">16</a>
To check the version of the ECU Software Porting Kit.	

## ds4121\_init

### Syntax

```
void ds4121_init(phs_addr_t base);
```

### Include file

ds4121.h

### Purpose

To initialize the DS4121.

### Parameters

**base** PHS-bus base address of the DS4121 board

### Description

The board is initialized as follows:

- The registers of the board are initialized.
- The FIFOs are reset.
- The interrupt inputs are set to edge triggered.

**Return value** None

**Messages** The following message is defined:

Type	Message	Meaning
Error	ds4121_init(): Invalid PHS-bus address 0x???????	The value of the base parameter is not a valid PHS-bus address. This error may be caused if the PHS-bus connection of the I/O board is missing. Check the connection.
Error	ds4121_init(): (0xXX): Board not found!	The DS4121 board was not found at the specified PHS-bus address. Check if the DSxxxx_n_BASE macro corresponds to the I/O board used.

**Related topics**

**References**

[DS4121\\_ECUSETUP\\_Bx \(DS4121 RTI Reference !\[\]\(c694a3ff3b077d76910920a6a1593ab4\_img.jpg\)](#))

## ds4121\_connection\_check

**Syntax**

```
UInt32 ds4121_connection_check(
    phs_addr_t base,
    UInt32 channel);
```

**Include file** ds4121.h

**Purpose** To check if an ECU is connected to the DS4121.

**Description** To check if an ECU is connected to the DS4121 at the given channel.

**ds4121\_connection\_check** generates a warning message when the connection has been lost. An info message is generated when the connection has been reestablished. Additionally, you can evaluate the return value to get the information.

**Parameters**

**base** PHS-bus base address of the DS4121 board

**channel** DS4121 channel number; the possible values are 1 or 2.

**Return value** Error code. The following macros are predefined:

Return Value	Meaning
DS4121_CONNECTION_OK	The connection between the DS4121 channel and the ECU has been established.
DS4121_CONNECTION_LOST	The connection between the DS4121 channel and the ECU has been lost. This can be caused by missing or faulty wiring or by an ECU without power supply.

**Messages** The following information messages are defined:

Type	Message	Meaning
Info	ds4121_connection_check(0xXX): Connection to ECU channel x established!	The DS4121 channel has established a connection to an ECU.
Warning	ds4121_connection_check(0xXX): No connection to ECU channel x!	The DS4121 channel could not establish a connection to an ECU. Check the physical connection between the DS4121 and the ECU and check the power supply of the ECU.

## Related topics

### Basics

[ECU Interface Unit \(DS4121 Features !\[\]\(830769b31eeeaca920791081939ff8ba\_img.jpg\)\)](#)

## ds4121\_fifo\_check

### Syntax

```
UInt32 ds4121_fifo_check(
    phs_addr_t base,
    UInt32 channel);
```

### Include file

ds4121.h

### Purpose

To check the state of the FIFO of the specified communication channel.

#### Note

The DS4121 write functions check the FIFO state automatically.

<b>Parameters</b>	<b>base</b> PHS-bus base address of the DS4121 board
	<b>channel</b> DS4121 channel number; the possible values are 1 or 2.

**Return value**                      Error code. The following macros are predefined:

Return Value	Meaning
DS4121_FIFO_NOT_FULL	The FIFO of the specified channel is not full.
DS4121_FIFO_ALMOST_FULL	There are less than 64 entries free in the FIFO of the specified channel.
DS4121_FIFO_FULL	The FIFO of the specified channel is full.

## Related topics

### Basics

[ECU Interface Unit \(DS4121 Features !\[\]\(5a132f13505a6571904d622757b7a8f0\_img.jpg\)\)](#)

### References

<a href="#">ds4121_fifo_reset.....</a>	<a href="#">14</a>
<a href="#">ds4121_fifo_start.....</a>	<a href="#">15</a>

## ds4121\_fifo\_reset

**Syntax**

```
void ds4121_fifo_reset(  
    phs_addr_t base,  
    UInt32 channel);
```

**Include file**                      `ds4121.h`

**Purpose**                              To reset the FIFO buffer of the specified communication channel.

<b>Parameters</b>	<b>base</b> PHS-bus base address of the DS4121 board
	<b>channel</b> DS4121 channel number; the possible values are 1 or 2.

**Return value**                      None

**Related topics****Basics**

[ECU Interface Unit \(DS4121 Features !\[\]\(feabb98897b440bc8695a03336a6e2df\_img.jpg\)\)](#)

**References**

[ds4121\\_fifo\\_check](#)..... 13  
[ds4121\\_fifo\\_start](#)..... 15

## ds4121\_fifo\_start

**Syntax**

```
void ds4121_fifo_start(
    phs_addr_t base,
    UInt32 channel);
```

**Include file**

ds4121.h

**Purpose**

To release the reset of the FIFO of the specified communication channel after calling `ds4121_fifo_reset`.

**Note**

For a reset of the FIFO, a period of 200 ns is required between `ds4121_fifo_reset` and `ds4121_fifo_start`.

**Parameters**

**base**    PHS-bus base address of the DS4121 board

**channel**    DS4121 channel number; the possible values are 1 or 2.

**Return value**

None

**Related topics****Basics**[ECU Interface Unit \(DS4121 Features !\[\]\(3d8c13c92b853674f749aac6fa869926\_img.jpg\)\)](#)**References**

<a href="#">ds4121_fifo_check</a> .....	13
<a href="#">ds4121_fifo_reset</a> .....	14

## ds4121\_ecu\_version\_check

**Syntax**

```
UInt32 ds4121_ecu_version_check(  
    phs_addr_t base,  
    UInt32 channel);
```

**Include file**

ds4121.h

**Purpose**

To check the version of dSPACE's ECU Software Porting Kit used by the ECU connected to the specified channel (refer to [ECU Software Porting Kit \(DS4121 Features !\[\]\(4688aadfd656ded00cd6bdfae55089a9\_img.jpg\)\)](#))).

**Note**

- ds4121\_subint\_init must be called before the ds4121\_ecu\_version\_check function can be used.
- Use this function only if your ECU supports the subinterrupts provided by dSPACE's ECU Software Porting Kit.

**Parameters****base** PHS-bus base address of the DS4121 board**channel** DS4121 channel number; the possible values are 1 or 2.**Return value**

Error code. The following macros are predefined:

Return Value	Meaning
DS4121_ECU_VERSION_OK	The ECU Software Porting Kit version found is compatible with the RTLib version used.
DS4121_ECU_VERSION_NOT_AVAILABLE	The ECU Software Porting Kit version could not be evaluated. This occurs when the ECU is not yet alive.



Return Value	Meaning
DS4121_ECU_VERSION_OBSOLETE	The ECU Software Porting Kit version found is old and not compatible with the RTLib version used. Use the current version of the ECU Software Porting Kit (contact <a href="mailto:support@dspace.de">support@dspace.de</a> or call the technical support team) to recompile and download your ECU code.
DS4121_ECU_WRONG_COMPILATION	dSPACE's ECU Software Porting Kit supports different ECU interface boards. The subinterrupt handling on your ECU has been compiled for another dSPACE board and is not compatible with the board used.

## Messages

The following messages are defined:

Type	Message	Meaning
Info	ds4121_ecu_version_check(0xXX): ECU Software Porting Kit rev. X.Y.ZZ detected on ch x!	Version X.Y.ZZ of the ECU Software Porting Kit was detected. This version is compatible with the RTLib version used.
Warning	ds4121_ecu_version_check(0xXX): ECU SW Vs. not yet available on ch x!	Normally, the ECU is powered up after the dSPACE system. In this case, the version of the ECU Software Porting Kit is not available yet. This information is displayed if <ul style="list-style-type: none"> <li>▪ The ECU is not running,</li> <li>▪ The ECU is powered up after the dSPACE system or</li> <li>▪ The ECU is not alive.</li> </ul>
Warning	ds4121_ecu_version_check(0xXX): ECU software rev. X.Y or higher required on ch x!	The ECU Software Porting Kit version found is old and not compatible with the RTLib version used. Use the current version of the ECU Software Porting Kit (contact <a href="mailto:support@dspace.de">support@dspace.de</a> or call the technical support team) to recompile and download your ECU code.
Warning	ds4121_ecu_version_check(0xXX): ECU software is not compiled for DS4121 on ch x!	dSPACE's ECU Software Porting Kit supports different ECU interface boards. The subinterrupt handling on your ECU has been compiled for another dSPACE board and is not compatible with the board used. For information on how to set up the ECU Software Porting Kit for a DS4121, refer to <a href="#">Generating the ECU Specific File</a> on page 36.

## Related topics

### Basics

[ECU Software Porting Kit \(DS4121 Features !\[\]\(003082e50e3009141f59bd5df831749f\_img.jpg\)](#))

### References

ds4121\_subint\_init..... 18  
[Generating the ECU Specific File](#)..... 36

# Interrupt Handling

## Where to go from here

## Information in this section

<a href="#">ds4121_subint_init</a> .....	18
To initialize the subinterrupt receiver	
<a href="#">ds4121_end_of_int_set</a> .....	19
To indicate the end of an interrupt	

## Information in other sections

[Interrupt Handling \(DS4121 Features !\[\]\(a870788d6ed9b8fd294b7654a8c8526b\_img.jpg\)](#))

## ds4121\_subint\_init

### Syntax

```
void ds4121_subint_init(
    phs_addr_t base,
    UInt32 channel,
    UInt32 nr_sints,
    UInt32 thirty_plus_x_free_dpm_words);
```

### Include file

ds4121.h

### Purpose

To create the subinterrupt receiver for the onboard DPMEM.

### Description

A subinterrupt receiver is initialized for the onboard DPMEM. The subinterrupt locations are initialized with "4", that means "ready invalid".

### Parameters

**base**     PHS-bus base address of the DS4121 board

**channel**     DS4121 channel number; the possible values are 1 or 2.

**nr\_sints**     Number of subinterrupts to be used within the range of 1 ... 16

**thirty\_plus\_x\_free\_dpm\_words**     Start address of  $30 + x$  (with  $x$  = number of subinterrupts) unused words in the onboard DPMEM that is allocated to the subinterrupt module. The usage of the words is defined in `dssint_wb_relation.h`.

**Return value** None

**Messages** The following messages are defined:

Type	Message	Meaning
Error	ds4121_subint_init (0xXX): Memory allocation error!	No memory could be allocated for the function.
Error	ds4121_subint_init (0xXX): Invalid channel number!	Select the ECU channel 1 or 2.

## Related topics

### Basics

[Interrupt Handling \(DS4121 Features !\[\]\(cbe2492b119e39e02a1dab2af4a4b296\_img.jpg\)\)](#)  
[Specifying DPMEM Addresses Seen from the ECU \(DS4121 Features !\[\]\(2f36c159ea3670f7a62f64a4f1cf5c05\_img.jpg\)\)](#)

### References

[DS4121\\_ECUINT\\_Bx\\_CHy\\_I0, DS4121\\_ECUINT\\_Bx\\_CHy\\_Siz \(DS4121 RTI Reference !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0\_img.jpg\)\)](#)

## ds4121\_end\_of\_int\_set

### Syntax

```
void ds4121_end_of_int_set(
    phs_addr_t base,
    UInt32 channel);
```

**Include file** ds4121.h

**Purpose** To indicate the end of an interrupt to the ECU.

**Parameters**

**base** PHS-bus base address of the DS4121 board

**channel** DS4121 channel number; the possible values are 1 or 2.

**Return value** None

## Related topics

### Basics

[Interrupt Handling \(DS4121 Features !\[\]\(dfbd6b3763a6d1d9afaa974f64e2e4b5\_img.jpg\)\)](#)

### References

[DS4121\\_ECUINT\\_Bx\\_CHy\\_I0, DS4121\\_ECUINT\\_Bx\\_CHy\\_Siz \(DS4121 RTI Reference !\[\]\(23d9fc146e83b5c3013cfa32c784f8d5\_img.jpg\)\)](#)

# Reading Data from the ECU

## Where to go from here

## Information in this section

<a href="#">ds4121_p_int_read8</a> .....	21
To read 8-bit integer data from the DPMEM (packed format).	
<a href="#">ds4121_s_int_read8</a> .....	22
To read 8-bit integer data from the DPMEM (scattered format).	
<a href="#">ds4121_le_p_int_read16, ds4121_le_p_int_read32, ds4121_le_s_int_read16, ds4121_le_s_int_read32</a> .....	23
To read 16- or 32-bit integer data from the DPMEM (little endian, packed or scattered format).	
<a href="#">ds4121_be_p_int_read16, ds4121_be_p_int_read32, ds4121_be_s_int_read16, ds4121_be_s_int_read32</a> .....	24
To read 16- or 32-bit integer data from the DPMEM (big endian, packed or scattered format).	
<a href="#">ds4121_le_p_fl_read32, ds4121_le_s_fl_read32, ds4121_be_p_fl_read32, ds4121_be_s_fl_read32</a> .....	25
To read 32-bit floating point data (IEEE Std. 754) from the DPMEM (big or little endian, packed or scattered format).	

## Information in other sections

<a href="#">DS4121 RTLib Functions</a> .....	8
To get an overview of the data types and the function names used.	

## ds4121\_p\_int\_read8

### Syntax

```
void ds4121_p_int_read8(
    phs_addr_t base,
    UInt32 channel ,
    UInt32 count,
    UInt32 *address,
    UInt32 *halfword,
    UInt8 *value);
```

### Include file

ds4121.h

### Purpose

To read 8-bit integer data from the DPMEM (packed format).

**Description**

For more information on data types, refer to [DS4121 RTLib Functions](#) on page 8 and [Supported ECU Data Type Formats \(DS4121 Features\)](#).

For information on the data arrangement in the DPMEM, refer to [Data Layout in the DPMEM](#) on page 9.

**Parameters**

**base** PHS-bus base address of the DS4121 board

**channel** DS4121 channel number; the possible values are 1 or 2.

**count** Number of values to be read

**address** Pointer to an array containing the addresses to be read (16-bit DPMEM addresses from the view of the RCP system)

**halfword** Pointer to an array containing the information for each address which halfword (8 bit) has to be read. The following macros are predefined:

Predefined Macros	Meaning
DS4121_LOWER_HALFWORD	To read the lower halfword from the related 16-bit DPMEM address.
DS4121_UPPER_HALFWORD	To read the upper halfword from the related 16-bit DPMEM address.

**value** Pointer to an array where the values of the specified addresses will be stored

**Return value**

None

**Related topics**

## References

[DS4121\\_ECUREAD\\_Bx\\_CHy\\_BLz \(DS4121 RTI Reference\)](#)



## ds4121\_s\_int\_read8

**Syntax**

```
void ds4121_s_int_read8(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UInt8 *value);
```

**Include file**

ds4121.h

<b>Purpose</b>	To read 8-bit integer data from the DPMEM (scattered format). The scattered format reads only from the lower halfword.
<b>Description</b>	<p>For more information on data types, refer to <a href="#">DS4121 RTLib Functions</a> on page 8 and <a href="#">Supported ECU Data Type Formats (DS4121 Features </a>).</p> <p>For information on the data arrangement in the DPMEM, refer to <a href="#">Data Layout in the DPMEM</a> on page 9.</p>
<b>Parameters</b>	<p><b>base</b> PHS-bus base address of the DS4121 board</p> <p><b>channel</b> DS4121 channel number; the possible values are 1 or 2.</p> <p><b>count</b> Number of values to be read</p> <p><b>address</b> Pointer to an array containing the addresses to be read (16-bit DPMEM addresses from the view of the RCP system)</p> <p><b>value</b> Pointer to an array where the values of the specified addresses will be stored</p>
<b>Return value</b>	None
<b>Related topics</b>	<p>References</p> <div> <a href="#">DS4121_ECUREAD_Bx_CHy_BLz (DS4121 RTI Reference </a>) </div>



[ds4121\\_le\\_p\\_int\\_read16](#), [ds4121\\_le\\_p\\_int\\_read32](#),  
[ds4121\\_le\\_s\\_int\\_read16](#), [ds4121\\_le\\_s\\_int\\_read32](#)

**Syntax**

```
void ds4121_le_y_int_readzz(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UIntzz *value);
```

**Include file**

ds4121.h

<b>Purpose</b>	To read 16-bit or 32-bit integer data from the DPMEM (little endian, scattered or packed format).
<b>Description</b>	<p>For more information on data types, refer to <a href="#">DS4121 RTLib Functions</a> on page 8 and <a href="#">Supported ECU Data Type Formats (DS4121 Features </a>).</p> <p>For information on the data arrangement in the DPMEM, refer to <a href="#">Data Layout in the DPMEM</a> on page 9.</p>
<b>Parameters</b>	<p><b>base</b> PHS-bus base address of the DS4121 board</p> <p><b>channel</b> DS4121 channel number; the possible values are 1 or 2.</p> <p><b>count</b> Number of values to be read</p> <p><b>address</b> Pointer to an array containing the addresses to be read (16-bit DPMEM addresses from the view of the RCP system)</p> <p><b>value</b> Pointer to an array where the values of the specified addresses will be stored</p>
<b>Return value</b>	None
<b>Related topics</b>	<p>References</p> <div><a href="#">DS4121_ECUREAD_Bx_CHy_BLz (DS4121 RTI Reference </a>)</div>

[ds4121\\_be\\_p\\_int\\_read16](#), [ds4121\\_be\\_p\\_int\\_read32](#),  
[ds4121\\_be\\_s\\_int\\_read16](#), [ds4121\\_be\\_s\\_int\\_read32](#)



**Syntax**

```
void ds4121_be_y_int_readzz(  
    phs_addr_t base,  
    UInt32 channel,  
    UInt32 count,  
    UInt32 *address,  
    UIntzz *value);
```

**Include file**

ds4121.h



<b>Purpose</b>	To read 16-bit or 32-bit integer data from the DPMEM (big endian, scattered or packed format).
<b>Description</b>	<p>For more information on data types, refer to <a href="#">DS4121 RTLib Functions</a> on page 8 and <a href="#">Supported ECU Data Type Formats (DS4121 Features </a>).</p> <p>For information on the data arrangement in the DPMEM, refer to <a href="#">Data Layout in the DPMEM</a> on page 9.</p>
<b>Parameters</b>	<p><b>base</b> PHS-bus base address of the DS4121 board</p> <p><b>channel</b> DS4121 channel number; the possible values are 1 or 2.</p> <p><b>count</b> Number of values to be read</p> <p><b>address</b> Pointer to an array containing the addresses to be read (16-bit DPMEM addresses from the view of the RCP system)</p> <p><b>value</b> Pointer to an array where the values of the specified addresses will be stored</p>
<b>Return value</b>	None
<b>Related topics</b>	<p>References</p> <div> <a href="#">DS4121_ECUREAD_Bx_CHy_BLz (DS4121 RTI Reference </a>) </div>

[ds4121\\_le\\_p\\_fl\\_read32](#), [ds4121\\_le\\_s\\_fl\\_read32](#), [ds4121\\_be\\_p\\_fl\\_read32](#), [ds4121\\_be\\_s\\_fl\\_read32](#)

**Syntax**

```
void ds4121_ob_dpm_xx_y_fl_read32(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UInt32 *value);
```


**Include file**

ds4121.h

---

<b>Purpose</b>	To read 32-bit floating point data from the DPMEM ( <i>big or little endian, scattered or packed format</i> ).
----------------	--

---

<b>Description</b>	<p>For more information on data types, refer to <a href="#">DS4121 RTLib Functions</a> on page 8 and <a href="#">Supported ECU Data Type Formats (DS4121 Features </a>).</p> <p>For information on the data arrangement in the DPMEM, refer to <a href="#">Data Layout in the DPMEM</a> on page 9.</p>
--------------------	---

---

<b>Parameters</b>	<p><b>base</b>    PHS-bus base address of the DS4121 board</p> <p><b>channel</b>    DS4121 channel number; the possible values are 1 or 2.</p> <p><b>count</b>    Number of values to be read</p> <p><b>address</b>    Pointer to an array containing the addresses to be read (16-bit DPMEM addresses from the view of the RCP system)</p> <p><b>value</b>    Pointer to an array where the values of the specified addresses will be stored</p>
-------------------	---

---

<b>Return value</b>	None
---------------------	------

---

**Related topics****References**

[DS4121\\_ECUREAD\\_Bx\\_CHy\\_BLz \(DS4121 RTI Reference !\[\]\(de95854c7ee024cfadc48187bbb781b2\_img.jpg\)](#))

# Transferring Data to the ECU

## Where to go from here

## Information in this section

<a href="#">ds4121_p_int_write8</a> .....	27
To write 8-bit integer data to the DPMEM (packed format).	
<a href="#">ds4121_s_int_write8</a> .....	29
To write 8-bit integer data to the DPMEM (scattered format).	
<a href="#">ds4121_le_p_int_write16, ds4121_le_p_int_write32, ds4121_le_s_int_write16, ds4121_le_s_int_write32</a> .....	30
To write 16- or 32-bit integer data to the DPMEM (little endian, packed or scattered format).	
<a href="#">ds4121_be_p_int_write16, ds4121_be_p_int_write32, ds4121_be_s_int_write16, ds4121_be_s_int_write32</a> .....	31
To write 16- or 32-bit integer data to the DPMEM (big endian, packed or scattered format).	
<a href="#">ds4121_le_p_fl_write32, ds4121_le_s_fl_write32, ds4121_be_p_fl_write32, ds4121_be_s_fl_write32</a> .....	32
To write 32-bit floating point data (IEEE Std. 754) to the DPMEM (big or little endian, packed or scattered format).	

## Information in other sections

<a href="#">DS4121 RTLib Functions</a> .....	8
To get an overview of the data types and the function names used.	

## ds4121\_p\_int\_write8

### Syntax

```
void ds4121_p_int_write8(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UInt32 *halfword,
    UInt8 *value);
```

### Include file

ds4121.h

### Purpose

To write 8-bit integer data to the DPMEM (packed format).

**Description**

For more information on data types, refer to [DS4121 RTLib Functions](#) on page 8 and [Supported ECU Data Type Formats \(DS4121 Features !\[\]\(2bdfe261b986065ee0ac76460d6528c9\_img.jpg\)](#)).

For information on the data arrangement in the DPMEM, refer to [Data Layout in the DPMEM](#) on page 9.

**Note**

This function uses a FIFO for the write transfer of the specified channel. If the FIFO is full, the function waits until there is enough space available for the data transfer. If the connection to the ECU has been lost during the transfer, the function blocks the application and causes an overload error.

**Parameters**

**base** PHS-bus base address of the DS4121 board

**channel** DS4121 channel number; the possible values are 1 or 2.

**count** Number of values to be written

**address** Pointer to an array containing the addresses to be written to (16-bit DPMEM addresses from the view of the RCP system)

**halfword** Pointer to an array containing the information for each address which halfword (8 bit) has to be written. The following macros are predefined:

Predefined Macros	Meaning
DS4121_LOWER_HALFWORD	To read the lower halfword from the related 16-bit DPMEM address.
DS4121_UPPER_HALFWORD	To read the upper halfword from the related 16-bit DPMEM address.

**value** Pointer to an array containing the values to be written

**Return value**

None

**Related topics****References**

[DS4121\\_ECUWRITE\\_Bx\\_ChY\\_BLz \(DS4121 RTI Reference !\[\]\(cbd8541a32dfc32f356f5c6c994b0a21\_img.jpg\)](#))

## ds4121\_s\_int\_write8

### Syntax

```
void ds4121_s_int_write8(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UInt8 *value);
```

### Include file

ds4121.h

### Purpose

To write 8-bit integer data to the DPMEM (scattered format). The scattered format writes only to the lower halfword.

### Description

For more information on data types, refer to [DS4121 RTLib Functions](#) on page 8 and [Supported ECU Data Type Formats \(DS4121 Features\)](#).

For information on the data arrangement in the DPMEM, refer to [Data Layout in the DPMEM](#) on page 9.

#### Note

This function uses a FIFO for the write transfer of the specified channel. If the FIFO is full, the function waits until there is enough space available for the data transfer. If the connection to the ECU has been lost during the transfer, the function blocks the application and causes an overload error.

### Parameters

**base** PHS-bus base address of the DS4121 board

**channel** DS4121 channel number; the possible values are 1 or 2.

**count** Number of values to be written

**address** Pointer to an array containing the DPMEM addresses to be written to (16-bit DPMEM addresses from the view of the RCP system)

**value** Pointer to an array containing the values to be written

### Return value

None

### Related topics

#### References

[DS4121\\_ECUWRITE\\_Bx\\_CHy\\_BLz \(DS4121 RTI Reference\)](#)

## ds4121\_le\_p\_int\_write16, ds4121\_le\_p\_int\_write32, ds4121\_le\_s\_int\_write16, ds4121\_le\_s\_int\_write32

### Syntax

```
void ds4121_le_y_int_writezz(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UIntzz *value);
```

### Include file

ds4121.h

### Purpose

To write 16-bit or 32-bit integer data (little endian, scattered or packed format) to the DPMEM.

### Description

For more information on data types, refer to [DS4121 RTLib Functions](#) on page 8 and [Supported ECU Data Type Formats \(DS4121 Features !\[\]\(73002692dd5e7a64e60946be3158e719\_img.jpg\)](#)).

For information on the data arrangement in the DPMEM, refer to [Data Layout in the DPMEM](#) on page 9.

#### Note

These functions use a FIFO for the write transfer of the specified channel. If the FIFO is full, the function waits until there is enough space available for the data transfer. If the connection to the ECU has been lost during the transfer, the function blocks the application and causes an overload error.

### Parameters

**base**    PHS-bus base address of the DS4121 board

**channel**    DS4121 channel number; the possible values are 1 or 2.

**count**    Number of values to be written

**address**    Pointer to an array containing the DPMEM addresses to be written to (16-bit DPMEM addresses from the view of the RCP system)

**value**    Pointer to an array containing the values to be written

### Return value

None

**Related topics****References**

[DS4121\\_ECUWRITE\\_Bx\\_CHy\\_BLz \(DS4121 RTI Reference !\[\]\(feabb98897b440bc8695a03336a6e2df\_img.jpg\)](#))

[ds4121\\_be\\_p\\_int\\_write16](#), [ds4121\\_be\\_p\\_int\\_write32](#),  
[ds4121\\_be\\_s\\_int\\_write16](#), [ds4121\\_be\\_s\\_int\\_write32](#)

**Syntax**

```
void ds4121_be_y_int_writezz(
    phs_addr_t base,
    UInt32 channel,
    UInt32 count,
    UInt32 *address,
    UIntzz *value);
```

**Include file**

ds4121.h

**Purpose**

To write 16-bit or 32-bit integer data (big endian, scattered or packed format) to the DPMEM.

**Description**

For more information on data types, refer to [DS4121 RTLib Functions](#) on page 8 and [Supported ECU Data Type Formats \(DS4121 Features !\[\]\(c444627dab9fee9a1550c053ffaaaae2\_img.jpg\)](#)).

For information on the data arrangement in the DPMEM, refer to [Data Layout in the DPMEM](#) on page 9.

**Note**

These functions use a FIFO for the write transfer of the specified channel. If the FIFO is full, the function waits until there is enough space available for the data transfer. If the connection to the ECU has been lost during the transfer, the function blocks the application and causes an overload error.

---

<b>Parameters</b>	<b>base</b> PHS-bus base address of the DS4121 board <b>channel</b> DS4121 channel number; the possible values are 1 or 2. <b>count</b> Number of values to be written <b>address</b> Pointer to an array containing the DPMEM addresses to be written to (16-bit DPMEM addresses from the view of the RCP system) <b>value</b> Pointer to an array containing the values to be written
-------------------	---

---

---

<b>Return value</b>	None
---------------------	------

---

---

<b>Related topics</b>	References
-----------------------	------------

[DS4121\\_ECUWRITE\\_Bx\\_CHy\\_BLz \(DS4121 RTI Reference !\[\]\(6605b201d6f14d9b3bcb8ab5f274d107\_img.jpg\)](#))

[ds4121\\_le\\_p\\_fl\\_write32](#), [ds4121\\_le\\_s\\_fl\\_write32](#),  
[ds4121\\_be\\_p\\_fl\\_write32](#), [ds4121\\_be\\_s\\_fl\\_write32](#)

---

<b>Syntax</b>	<pre>void ds4121_xx_y_fl_write32(     phs_addr_t base,     UInt32 channel,     UInt32 count,     UInt32 *address,     UInt32 *value);</pre>
---------------	---

---

---

<b>Include file</b>	ds4121.h
---------------------	----------


---

---

<b>Purpose</b>	To write 32-bit floating point data ( <i>big</i> or <i>little</i> endian, scattered or packed format) to the DPMEM.
----------------	---

---

---

<b>Description</b>	For more information on data types, refer to <a href="#">DS4121 RTLib Functions</a> on page 8 and <a href="#">Supported ECU Data Type Formats (DS4121 Features </a> ).
--------------------	---

---



For information on the data arrangement in the DPMEM, refer to [Data Layout in the DPMEM](#) on page 9.

**Note**

These functions use a FIFO for the write transfer of the specified channel. If the FIFO is full, the function waits until there is enough space available for the data transfer. If the connection to the ECU has been lost during the transfer, the function blocks the application and causes an overload error.

<b>Parameters</b>	<p><b>base</b>    PHS-bus base address of the DS4121 board</p> <p><b>channel</b>    DS4121 channel number; the possible values are 1 or 2.</p> <p><b>count</b>    Number of values to be written</p> <p><b>address</b>    Pointer to an array containing the DPMEM addresses to be written to (16-bit DPMEM addresses from the view of the RCP system)</p> <p><b>value</b>    Pointer to an array containing the values to be written</p>
-------------------	---

<b>Return value</b>	None
---------------------	------

<b>Related topics</b>	<p>References</p> <p><a href="#">DS4121_ECUWRITE_Bx_CHy_BLz</a> (DS4121 RTI Reference )</p>
-----------------------	--



# ECU Software Porting Kit

**Introduction**                      The ECU Software Porting Kit allows you to split one ECU-to-RTP interrupt into 16 subinterrupts (0 ... 15).  
For further information, refer to [ECU Software Porting Kit \(DS4121 Features !\[\]\(c507f772dba2b921f86777f01218e570\_img.jpg\)](#)).

Where to go from here	Information in this section
	Generating the ECU Specific File.....36
	Initializing the ECU Software Porting Kit.....37
	Subinterrupt Programming.....45
	Host Utilities.....50

## Generating the ECU Specific File

### Defines to be Specified in the dsECUframe.c File

#### Required defines

The following defines have to be specified in the ECU-specific `dsECUframe.c` file:

Define	Description
<code>_DSECU</code>	To identify the ECU target type.  <b>NOTICE</b> <b>Do not modify this define.</b>
<code>DSECU_INT16</code>	To specify the integer data type to be used for a width of 16 bit. Possible values are: short, int or long.
<code>DSECU_THIRTY_PLUS_X_FREE_DPM_WORDS</code>	To specify the start address for $30 + x$ (with $x$ = number of subinterrupts to be used) successive DPMEM addresses to be used for subinterrupt handling.
<code>DSECU_ECU2RTP_INT</code>	To specify the DPMEM address. When the ECU writes to this location, an interrupt is triggered on the RTP.
<code>DSECU_COMPILATION</code>	To specify the dSPACE ECU interface board. The specification is used for the compilation process: Depending on the specified board, the compilation results in different interrupt triggering DPMEM address. You can combine the predefined macros via the logical operator OR. The following macros are predefined: <ul style="list-style-type: none"> <li>▪ <code>DSSINT_WB_COMPILED_FOR_DS4121</code> Compilation for a DS4121 board.</li> <li>▪ <code>DSSINT_WB_COMPILED_FOR_DS1401</code> Compilation for an ECU module of a MicroAutoBox II.</li> <li>▪ <code>DSSINT_WB_COMPILED_FOR_INDEPENDENT</code> Compilation independent of the hardware.</li> </ul>

#### Related topics

#### References

[dsECUnew.bat](#)..... 50

# Initializing the ECU Software Porting Kit

**Introduction** To use subinterrupt handling for your ECU, you have to initialize the subinterrupt module.

## Where to go from here

## Information in this section

<a href="#">DSECU_SINT_INIT.....</a>	<a href="#">37</a>
To simplify the initialization of the subinterrupt module.	
<a href="#">dsecu_define_sender.....</a>	<a href="#">39</a>
To initialize the subinterrupt module.	
<a href="#">dsecu_startup.....</a>	<a href="#">40</a>
To start the alive mechanism.	
<a href="#">dsecu_alive.....</a>	<a href="#">41</a>
To continue the alive mechanism and check if the RTP is alive.	
<a href="#">dsecu_sender_suspend.....</a>	<a href="#">43</a>
To suspend the alive mechanism.	

## Information in other sections

[ECU Software Porting Kit \(DS4121 Features !\[\]\(830769b31eeeaca920791081939ff8ba\_img.jpg\)\)](#)  
Word-Based Subinterrupt Handling

## DSECU\_SINT\_INIT

### Syntax

```
DSECU_SINT_INIT();
```

### Include file

dsECU.h

### Purpose

To simplify the initialization of the subinterrupt module.

### Description

This macro performs the following steps:

- Calls `dsecu_define_sender` to initialize the subinterrupt module for your specific ECU. The parameters are derived from the defines in **Defines to be**

Specified in the `dsECUframe.c` File. The relation between the parameters and the macros is as follows:

Parameter	Macro
sender	dssint_sender
nr_subinterrupts	DSECU_SINT_NUMBER
dpm_block_start_addr	DSECU_THIRTY_PLUS_X_FREE_DPM_WORDS
send_addr	DSECU_ECU2RTP_INT

- Writes the version information to the DPMEM.
- Calls `dsecu_startup` to start the alive mechanism (refer to [ECU Software Porting Kit \(DS4121 Features !\[\]\(756219e9389f679d57027482aa5cf5fc\_img.jpg\)](#))).

**Return value** None

### Example

```
void main(void)
{
    int sint_number;
    DSECU_SINT_INIT();
    while(1)
    {
        sint_number = rand() & 0x000f;
        DSECU_SINT_SEND(sint_number);
        if (DSECU_EOSI_POLL(sint_number) == DSECU_BYPASS_DATA_VALID)
        {
            read_bypass_data();
        }
    }
}
```

### Related topics

#### Basics

[ECU Software Porting Kit \(DS4121 Features !\[\]\(73002692dd5e7a64e60946be3158e719\_img.jpg\)](#))  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(42837a1907e26cf155e215b5440e265d\_img.jpg\)](#))

#### References

[Defines to be Specified in the dsECUframe.c File.....](#) 36  
[dsecu\\_define\\_sender.....](#) 39  
[dsecu\\_startup.....](#) 40

## dsecu\_define\_sender

### Syntax

```
void dsecu_define_sender(
    dsecu_sender_type *sender,
    unsigned int nr_subinterrupts,
    unsigned long dpm_block_start_addr,
    unsigned long send_addr);
```

### Include file

dsECU.h

### Purpose

To initialize a subinterrupt handler.

#### Note

It is recommended to use the macro DSECU\_SINT\_INIT that provides the necessary parameters automatically.

### Parameters

**sender** Address of the subinterrupt handler

**nr\_subinterrupts** Number of subinterrupts to be used in the application within the range of 1 ... 16

**dpm\_block\_start\_addr** Specifies the start address for  $30 + x$  (with  $x$  = number of subinterrupts to be used) successive DPMEM addresses to be used for subinterrupt handling.

**send\_addr** DPMEM address; if the ECU writes to this location an interrupt will be triggered on the RTP.

### Return value

None

### Example

This example initializes a subinterrupt handler via the standard macros defined in Defines to be Specified in the dsECUframe.c File.

```
dsecu_define_sender(
    &dssint_sender,
    DSECU_SINT_NUMBER,
    DSECU_30_PLUS_X_FREE_DPM_WORDS,
    DSECU_ECU2RTP_INT);
```

**Related topics****Basics**

[ECU Software Porting Kit \(DS4121 Features !\[\]\(3d8c13c92b853674f749aac6fa869926\_img.jpg\)](#))  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(ce455c990c00145a2dda1d9a310cb682\_img.jpg\)](#))

**References**

[DSECU\\_SINT\\_INIT.....37](#)

## dsecu\_startup

**Syntax**

```
void dsecu_startup(dsecu_sender_type *sender);
```

**Include file**

dsECU.h

**Purpose**

To start the alive mechanism (refer to [ECU Software Porting Kit \(DS4121 Features !\[\]\(4688aadfd656ded00cd6bdfae55089a9\_img.jpg\)](#))).

**Note**

This function must be called once when starting an application and after a suspension via `dsecu_sender_suspend`.

**Parameters**

**sender**    Address of the subinterrupt handler

**Return value**

None



## Example

```
void main(void)
{
    int startup_flag = 1;
    dsecu_startup(sender);
    while(1) /* background loop */
    {
        if (*ecu_suspend)
        {
            dsecu_sender_suspend(dssint_sender);
            startup_flag = 0;
        }
        else /* if ECU is not suspended */
        {
            if (!startup_flag)
            {
                dsecu_startup(&dssint_sender);
                startup_flag = 1;
            }
            dsecu_alive(&dssint_sender);
            ...
        }
    }
}
```

## Related topics

### Basics

[ECU Software Porting Kit \(DS4121 Features !\[\]\(d66ff64371a51729ac8c1cdaa685ba6f\_img.jpg\)\)](#)  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(0f31ebba7abcd47777e178db26f29705\_img.jpg\)\)](#)

### References

[dsecu\\_alive.....41](#)  
[dsecu\\_sender\\_suspend.....43](#)

# dsecu\_alive

## Syntax

```
void dsecu_alive(dsecu_sender_type *sender);
```

## Include file

dsECU.h

**Purpose**

To continue the alive mechanism and check if the partner system is still alive (refer to [ECU Software Porting Kit \(DS4121 Features !\[\]\(3dfb8d66e81160ad61421a3452093d1b\_img.jpg\)](#))).

**Note**

This function has to be called repetitively in the background loop as long as the connection from the ECU to the RTP is valid.

**Parameters**

**sender** Address of the subinterrupt handler

**Return value**

None

**Example**

```
void main(void)
{
    int startup_flag = 1;
    dsecu_startup(sender);
    while(1) /* background Loop*/
    {
        if (*ecu_suspend)
        {
            dsecu_sender_suspend(&dssint_sender);
            startup_flag = 0;
        }
        else /* if ECU is not suspended */
        {
            if (!startup_flag)
            {
                dsecu_startup(&dssint_sender);
                startup_flag = 1;
            }
            dsecu_alive(&dssint_sender);
            ...
        }
    }
}
```

**Related topics****Basics**

[ECU Software Porting Kit \(DS4121 Features !\[\]\(e3275251d0893157c3584e20c81dc3ba\_img.jpg\)](#))  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(9ab0e0ed3a1c2d865b438a931465ce60\_img.jpg\)](#))

**References**

[dsecu\\_sender\\_suspend.....](#) 43  
[dsecu\\_startup.....](#) 40

## dsecu\_sender\_suspend

### Syntax

```
void dsecu_sender_suspend(dsecu_sender_type sender);
```

### Include file

dsECU.h

### Purpose

To signal to the dSPACE system that the ECU is no longer alive.

### Parameters

**sender** Subinterrupt handler

### Return value

None

### Example

```
void main(void)
{
    int startup_flag = 1;
    dsecu_startup(sender);
    while(1) /* background Loop*/
    {
        if (*ecu_suspend)
        {
            dsecu_sender_suspend(dssint_sender);
            startup_flag = 0;
        }
        else /* if ECU is not suspended */
        {
            if (!startup_flag)
            {
                dsecu_startup(&dssint_sender);
                startup_flag = 1;
            }
            dsecu_alive(&dssint_sender);
            ...
        }
    }
}
```

Related topics

Basics

- [ECU Software Porting Kit \(DS4121 Features !\[\]\(633dd45d48d71eb51a85c6dd83ee51e9\_img.jpg\)](#))
- [Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(bdddf9191a284aa0945448444083c5b0\_img.jpg\)](#))

References

<a href="#">dsecu_alive.....</a>	<a href="#">41</a>
<a href="#">dsecu_startup.....</a>	<a href="#">40</a>

# Subinterrupt Programming

## Introduction

Subinterrupt handling allows you to split the hardware interrupt into 16 subinterrupts.

For more information on the interrupt handling, refer to *Interrupt Handling* and *Word-Based Subinterrupt Handling* in the *DS4121 Features* document.

## Where to go from here

## Information in this section

<a href="#">DSECU_SINT_SEND.....</a>	<a href="#">45</a>
To simplify the sending of a subinterrupt.	
<a href="#">dsecu_sint_send.....</a>	<a href="#">46</a>
To send a subinterrupt.	
<a href="#">DSECU_EOSI_POLL.....</a>	<a href="#">47</a>
To simplify the polling for the end-of-subinterrupt message.	
<a href="#">dsecu_eosi_poll.....</a>	<a href="#">48</a>
To poll the end-of-subinterrupt message until the RTP acknowledges the subinterrupt.	

## DSECU\_SINT\_SEND

### Syntax

```
DSECU_SINT_SEND(subinterrupt_number);
```

### Include file

dsECU.h

### Purpose

To simplify the sending of a subinterrupt.

### Description

This macro performs the following steps:

- Calls `dsecu_sint_send` to send a subinterrupt.
- Calls `dsecu_alive` to start the alive mechanism.

### Parameters

**subinterrupt\_number** Enter the number of the subinterrupt to be sent within the range of 0 ... 15

---

<b>Return value</b>	None
---------------------	------

---

**Example**

```
void main(void)
{
    int sint_number;
    DSECU_SINT_INIT();
    while(1)
    {
        sint_number = rand() & 0x000f;
        write_bypass_data(sint_number);
        DSECU_SINT_SEND(sint_number);
        if (DSECU_EOSI_POLL(sint_number) == DSECU_BYPASS_DATA_VALID)
        {
            read_bypass_data(sint_number);
        }
    }
}
```

**Related topics****Basics**

[ECU Software Porting Kit \(DS4121 Features !\[\]\(5a132f13505a6571904d622757b7a8f0\_img.jpg\)\)](#)  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(0f17417dd77a61b2fdbff69a33adf9f2\_img.jpg\)\)](#)

**References**

<a href="#">dsecu_alive.....</a>	<a href="#">41</a>
<a href="#">dsecu_sint_send.....</a>	<a href="#">46</a>

## dsecu\_sint\_send

**Syntax**

```
int dsecu_sint_send(
    dsecu_sender_type* sender,
    unsigned int sub_interrupt);
```

---



<b>Include file</b>	dsECU.h
---------------------	---------

---

<b>Purpose</b>	To send a subinterrupt.
----------------	-------------------------

**Note**

It is recommended to use the macro DSECU\_SINT\_SEND that provides the necessary parameters automatically.

<b>Parameters</b>	<p><b>sender</b>    Address of the subinterrupt handler</p> <p><b>sub_interrupt</b>    Enter the number of the subinterrupt to be sent within the range of 0 ... 15</p>						
<b>Return value</b>	<p>Error code; the following macros are predefined:</p> <table border="1"> <thead> <tr> <th>Return Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>DSECU_INTERRUPT_SENT</td><td>The subinterrupt was sent successfully.</td></tr> <tr> <td>DSECU_INTERRUPT_NOT_SENT</td><td>The subinterrupt could not be sent.</td></tr> </tbody> </table>	Return Value	Meaning	DSECU_INTERRUPT_SENT	The subinterrupt was sent successfully.	DSECU_INTERRUPT_NOT_SENT	The subinterrupt could not be sent.
Return Value	Meaning						
DSECU_INTERRUPT_SENT	The subinterrupt was sent successfully.						
DSECU_INTERRUPT_NOT_SENT	The subinterrupt could not be sent.						
<b>Related topics</b>	<p>Basics</p> <div> <a href="#">ECU Software Porting Kit (DS4121 Features )</a>  <a href="#">Word-Based Subinterrupt Handling (DS4121 Features )</a> </div>						

## DSECU\_EOSI\_POLL

<b>Syntax</b>	<code>DSECU_EOSI_POLL(subinterrupt_number);</code>
<b>Include file</b>	<code>dsECU.h</code>
<b>Purpose</b>	To simplify the polling for the end-of-subinterrupt message. This function sets the parameter <b>sender</b> automatically.
<b>Parameters</b>	<p><b>subinterrupt_number</b>    Number of the subinterrupt within the range of 0 ... 15</p>
<b>Return value</b>	None
<b>Example</b>	<code>void main(void)</code>

```

{
    int sint_number;
    DSECU_SINT_INIT();
    while(1)
    {
        sint_number = rand() & 0x000f;
        DSECU_SINT_SEND(sint_number);
        if (DSECU_EOSI_POLL(sint_number) == DSECU_BYPASS_DATA_VALID)
        {
            read_bypass_data();
        }
    }
}

```

## Related topics

### Basics

[ECU Software Porting Kit \(DS4121 Features !\[\]\(6605b201d6f14d9b3bcb8ab5f274d107\_img.jpg\)\)](#)  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(f4056bb2e5acf0a782fb9d812dad489d\_img.jpg\)\)](#)

### References

[dsecu\\_eosi\\_poll](#)..... 48

# dsecu\_eosi\_poll

## Syntax

```

int dsecu_eosi_poll(
    dsecu_sender_type sender,
    int sint_number);

```

## Include file

dsECU.h

## Purpose

To poll the end-of-subinterrupt message until the RTP acknowledges the subinterrupt, that means that the bypass data are valid.

### Note

Use the DSECU\_EOSI\_POLL macro, because it provides the necessary parameters automatically.

## Parameters

**sender**     Address of the subinterrupt handler

**sint\_number**     Number of the subinterrupt within the range of 0 ... 15



**Return value**

Error code; the following macros are predefined:

Return Value	Meaning
DSECU_BYPASS_DATA_VALID	The bypass data are valid.
DSECU_BYPASS_DATA_INVALID	The bypass data are invalid.

**Example**

```
if (DSECU_BYPASS_DATA_VALID == dsecu_eosi_poll(dssint_sender,sint))
{
    read_data();
}
```

**Related topics****Basics**

[ECU Software Porting Kit \(DS4121 Features !\[\]\(17413706fd4997a1a4bdf85c6864eee1\_img.jpg\)\)](#)  
[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(f419710cbe076aa30a9c6c031b5cbe84\_img.jpg\)\)](#)

**References**

[DSECU\\_EOSI\\_POLL.....47](#)

## Host Utilities

### dsECUnew.bat

---

**Purpose**

To create the new working directory and copy `dsECUframe.c` to the new directory.

---

**Description**

To work with the ECU Software Porting Kit, you have to create the working directory for the ECU application and copy the **Defines to be Specified in the dsECUframe.c File** file to this directory using the `dsECUnew.bat` function.

For already existing ECU projects, copy the `dsECUframe.c` file to the working directory.

Some ECU-specific settings must be made in the **Defines to be Specified in the dsECUframe.c File** file for the specific ECU used for your application. This file has to be included into your ECU build project. A template of the file is located in `<RCP_HIL_InstallationPath>\dsECU\PortingKit`.

---

**Syntax**

`dsECUnew Directory`

**Location**    `<RCP_HIL_InstallationPath>\Exe`

---

**Info messages**

The following information message is defined:

Type	Message	Description
Info	The ECU project "<projectname>" contains a <code>dsecuframe.c</code> file yet. The existing file will be opened.	The directory already exists and there is a file <code>dsECUframe.c</code> in the directory. The file will be opened.

---

**Related topics****References**

[Defines to be Specified in the dsECUframe.c File.....](#) 36

# Word-Based Subinterrupt Handling

Introduction

The Word-Based Subinterrupt module provides functions to extend one hardware interrupt to multiple software subinterrupts.

Where to go from here	Information in this section
	<div><div>dssint_wb_define_receiver..... 52</div><div>To define an interrupt receiver.</div><div>dssint_wb_subint_disable..... 53</div><div>To disable subinterrupts.</div><div>dssint_wb_subint_enable..... 54</div><div>To enable subinterrupts.</div><div>dssint_wb_decode..... 55</div><div>To find out which subinterrupts are pending.</div><div>dssint_wb_acknowledge..... 56</div><div>To acknowledge pending subinterrupts.</div><div>dssint_wb_subint_reset..... 57</div><div>To clear pending subinterrupts.</div><div>dssint_wb_end_of_sint_set..... 58</div><div>To finish a pending subinterrupt.</div><div>dssint_wb_startup..... 58</div><div>To start the alive mechanism.</div><div>dssint_wb_alive..... 60</div><div>To signal the <i>alive</i> state to the partner system.</div><div>dssint_wb_suspend..... 61</div><div>To signal the <i>not alive</i> state to the partner system.</div></div>

## dssint\_wb\_define\_receiver

### Syntax

```
dssint_wb_receiver_type *dssint_wb_define_receiver(
    long target,
    unsigned int nr_subinterrupts,
    unsigned long dpm_block_start_addr,
    unsigned long receiver_addr,
    dssint_wb_write_fcn_t write_fcn,
    dssint_wb_read_fcn_t read_fcn,
    dssint_wb_read_block_fcn_t read_block_fcn);
```

### Include file

Dssint\_wb.h

### Purpose

To create a receiver handler for the word-based subinterrupt handling.

### Description

The function reads from the **receiver\_addr** to enable interrupt triggering by the sender. It defines an interrupt receiver and returns a handle to it. The handle identifies the appropriate subinterrupt vector and receiving information table for a specific sender.

### Parameters

**target** Address of the target memory, for example, a PHS bus address or COM port number

**nr\_subinterrupts** Number of subinterrupts to be used in the application within the range of 1 ... 16

**dpm\_block\_start\_addr** Specifies the start address for  $30 + x$  (with  $x$  = number of subinterrupts to be used) successive DPMEM addresses to be used for subinterrupt handling.

**receiver\_addr** Memory location where the acknowledgment information from the receiver is passed

**write\_fcn** Address of a function that performs a write access to the dual-port memory

**read\_fcn** Address of a function that performs a read access to the dual-port memory

**read\_block\_fcn** Address of a function that performs a block read access to the dual-port memory

### Return value

Address of an interrupt receiver. The function returns "0" if an error has occurred.

**Example**

```
ds4121_subint_receiver[int_line][base_index] =
    (dssint_wb_receiver_type*)dssint_wb_define_receiver(
        base,
        nr_sints,
        thirty_plus_x_free_dpm_words,
        acknowledge_addr,
        (dssint_wb_write_fcn_t)write_fcn,
        (dssint_wb_read_fcn_t)read_fcn,
        (dssint_wb_read_block_fcn_t)read_block_fcn);
```

**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(74d4806277d7e73349d8e8c0897931e9\_img.jpg\)\)](#)

## dssint\_wb\_subint\_disable

**Syntax**

```
void dssint_wb_subint_disable(
    dssint_wb_receiver_type *receiver,
    unsigned int sub_interrupt);
```

**Include file**

Dssint\_wb.h

**Purpose**

To disable a subinterrupt.

**Description**

After initialization all subinterrupts are enabled. You must disable the subinterrupt explicitly via this function.

**Parameters**

**receiver**    Receiver handler where the subinterrupt is located in.  
**sub\_interrupt**    Subinterrupt to be disabled within the range of 0 ... 15

**Return value**

None

**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(eafc244b53721dd1ec133f0772f70fc7\_img.jpg\)\)](#)

**References**

[dssint\\_wb\\_subint\\_enable.....](#) 54  
[dssint\\_wb\\_subint\\_reset.....](#) 57

## dssint\_wb\_subint\_enable

**Syntax**

```
void dssint_wb_subint_enable(
    dssint_wb_receiver_type *receiver,
    unsigned int sub_interrupt);
```

**Include file**

Dssint\_wb.h

**Purpose**

To enable a subinterrupt.

**Description**

After initialization all subinterrupts are enabled. You can use this function to enable a subinterrupt after it has been disabled via `dssint_wb_subint_disable`.

**Parameters**

**receiver**    Receiver handler where the subinterrupt is located in.  
**sub\_interrupt**    Subinterrupt to be enabled within the range of 0 ... 15.

**Return value**

None

**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(6befd466863f06afb75445d91429f055\_img.jpg\)\)](#)

**References**

[dssint\\_wb\\_subint\\_disable.....](#) 53  
[dssint\\_wb\\_subint\\_reset.....](#) 57

## dssint\_wb\_decode

<b>Syntax</b>	<pre>int dssint_wb_decode(     dssint_wb_receiver_type *receiver);</pre>
<b>Include file</b>	Dssint_wb.h
<b>Purpose</b>	To find out which interrupts are pending.
<b>Description</b>	This function is called repetitively within an interrupt handler. It processes the interrupt information of the receiver data structure that was given by <b>dssint_wb_acknowledge</b> , determines the pending subinterrupt with the highest priority and returns it to the handler. The pending subinterrupt with highest priority is the one with the smallest subinterrupt number.
<b>Parameters</b>	<b>receiver</b> Receiver handler where the subinterrupt is located in.
<b>Return value</b>	Number of the pending subinterrupt with highest priority. If there is no pending subinterrupt left, the function returns SINT_WB_NO_SUBINT ("−1").

### Example

```
void sint_isr(void)
{
    int sint_number;
    dssint_wb_acknowledge(RECEIVER);
    do
    {
        sint_number = dssint_wb_decode(RECEIVER);
        if (sint_number > -1)
        {
            sint_counter[sint_number]++;
            dssint_wb_end_of_sint_set(RECEIVER, sint_number);
        }
    } while(sint_number > -1);
}
```

**Related topics****Basics**[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(3d8c13c92b853674f749aac6fa869926\_img.jpg\)\)](#)**References**

<a href="#">dssint_wb_acknowledge</a> .....	56
<a href="#">dssint_wb_end_of_sint_set</a> .....	58

## dssint\_wb\_acknowledge

**Syntax**

```
void dssint_wb_acknowledge(  
    dssint_wb_receiver_type *receiver);
```

**Include file**

Dssint\_wb.h

**Purpose**

To acknowledge all pending subinterrupts.

**Parameters****receiver**    Receiver handler where the subinterrupt is located in.**Return value**

None

**Example**

```
void sint_isr(void)  
{  
    int sint_number;  
    dssint_wb_acknowledge(RECEIVER);  
    do  
    {  
        sint_number = dssint_wb_decode(RECEIVER);  
        if (sint_number > -1)  
        {  
            sint_counter[sint_number]++;  
            dssint_wb_end_of_sint_set(RECEIVER, sint_number);  
        }  
    } while(sint_number > -1);  
}
```



**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(666e09182d4cd268646ea700ea60dcdf\_img.jpg\)\)](#)

**References**

[dssint\\_wb\\_decode.....](#) 55  
[dssint\\_wb\\_end\\_of\\_sint\\_set.....](#) 58

## dssint\_wb\_subint\_reset

**Syntax**

```
void dssint_wb_subint_reset(
    dssint_wb_receiver_type *receiver,
    unsigned int sub_interrupt);
```

**Include file**

Dssint\_wb.h

**Purpose**

To clear a pending subinterrupt.

**Parameters**

**receiver**    Receiver handler where the subinterrupt is located in.  
**sub\_interrupt**    Subinterrupt to be enabled within the range of 0 ... 15.

**Return value**

None

**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(19d44b37fb4fa155bf9d60c77a3d3cb2\_img.jpg\)\)](#)

**References**

[dssint\\_wb\\_subint\\_disable.....](#) 53  
[dssint\\_wb\\_subint\\_enable.....](#) 54

## dssint\_wb\_end\_of\_sint\_set

---

**Syntax**

```
void dssint_wb_end_of_sint_set(  
    dssint_wb_receiver_type *receiver,  
    unsigned int sub_interrupt);
```

---

**Include file**Dssint\_wb.h

---

**Purpose**To finish a pending subinterrupt.

---

**Parameters**

**receiver**    Receiver handler where the subinterrupt is located in.

**sub\_interrupt**    Subinterrupt to be enabled within the range of 0 ... 15.

---

**Return value**None

---

**Related topics****Basics**[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(f60b7a900783ac3fd531bfd9c111be6d\_img.jpg\)\)](#)

---

**References**

<a href="#">dssint_wb_acknowledge.....</a>	<a href="#">56</a>
<a href="#">dssint_wb_decode.....</a>	<a href="#">55</a>

---

## dssint\_wb\_startup

---


**Syntax**

```
void dssint_wb_startup(  
    dssint_wb_type *handler);
```

---

**Include file**Dssint\_wb.h


---

Purpose	To start the alive mechanism (refer to <a href="#">ECU Software Porting Kit (DS4121 Features </a> )).
	<div><b>Note</b> This function must be called once when starting an application and after a suspension via <code>dssint_wb_suspend</code>.</div>

Parameters	<b>handler</b> Handler where the subinterrupt receiver is located in.
------------	---

Return value	None
--------------	------

Example	<pre>void main(void) {     int startup_flag = 1;     dssint_wb_startup(RECEIVER);     ...     while(1) /* background Loop */     {         if (!suspend_flag)         {             if (!startup_flag)             {                 dssint_wb_startup(RECEIVER);                 startup_flag = 1;             }             dssint_wb_alive(RECEIVER);         }         else         {             dssint_wb_suspend(RECEIVER);             startup_flag = 0;         }         ...     } }</pre>
---------	--

Related topics	<div>Basics</div> <div>Word-Based Subinterrupt Handling (DS4121 Features </div>
	<div>References</div> <div><div>dssint_wb_alive..... 60</div><div>dssint_wb_suspend..... 61</div></div>

## dssint\_wb\_alive

### Syntax

```
void dssint_wb_alive(dssint_wb_type *handler);
```

### Include file

Dssint\_wb.h

### Purpose

To continue the alive mechanism and check if the partner system is still alive (refer to [ECU Software Porting Kit \(DS4121 Features !\[\]\(ec9132f1d27c8919987d92907322654d\_img.jpg\)](#))).

#### Note

This function has to be called repetitively in the background loop as long as the connection from the ECU to the RTP is valid.

### Parameters

**handler**    Handler where the subinterrupt receiver is located in.

### Return value

None

### Example

```
void main(void)
{
    int startup_flag = 1;
    dssint_wb_startup(RECEIVER);
    ...
    while(1) /* background loop */
    {
        if (!suspend_flag)
        {
            if (!startup_flag)
            {
                dssint_wb_startup(RECEIVER);
                startup_flag = 1;
            }
            dssint_wb_alive(RECEIVER);
        }
        else
        {
            dssint_wb_suspend(RECEIVER);
            startup_flag = 0;
        }
        ...
    }
}
```

**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(bd1a142de767a21e5362c595f844a4ff\_img.jpg\)\)](#)

**References**

[dssint\\_wb\\_startup..... 58](#)  
[dssint\\_wb\\_suspend..... 61](#)

## dssint\_wb\_suspend

**Syntax**

```
void dssint_wb_suspend(dssint_wb_type *handler);
```

**Include file**

Dssint\_wb.h

**Purpose**

To signal to the dSPACE system that the ECU is no longer alive.

**Parameters**

**handler**    Handler where the subinterrupt receiver is located in.

**Return value**

None

**Example**

```
void main(void)
{
    int startup_flag = 1;
    ...
    dssint_wb_startup(RECEIVER);
    ...
    while(1) /* background loop */
    {
        if (!suspend_flag)
        {
            if (!startup_flag)
            {
                dssint_wb_startup(RECEIVER);
                startup_flag = 1;
            }
            dssint_wb_alive(RECEIVER);
        }
        else
        {
            dssint_wb_suspend(RECEIVER);
            startup_flag = 0;
        }
        ...
    }
}
```

**Related topics****Basics**

[Word-Based Subinterrupt Handling \(DS4121 Features !\[\]\(950a62bbddad88d64435fd35607dfc42\_img.jpg\)](#))

**References**

[dssint\\_wb\\_alive..... 60](#)

# Function Execution Times

## Function Execution Times

### Introduction

Function execution times for DS4121 are measured with a standardized test environment.

### Test environment

The execution time of a function can vary, since it depends on different factors, for example:

- CPU clock and bus clock frequency of the processor board used
- Optimization level of the compiler
- Use of inlining parameters

The test programs that are used to measure the execution time of the functions listed below have been generated and compiled with the default settings of the `down<xxxx>` tool (optimization and inlining). The execution times in the tables below are always the mean measurement values.

The properties of the processor boards used are:

	<b>DS1006</b>	<b>DS1006 Multicore</b>
CPU clock	2.6 GHz / 3.0 GHz	2.8 GHz
Bus clock	133 MHz	133 MHz

### Overview

Execution times of the following RTLib functions are listed below.

- System functions
- Functions for reading data from the ECU
- Functions for transferring data to the ECU

### System functions

The following execution times have been measured for system functions (functions for initializing the board, accessing the FIFO and checking the ECU connection):

Function	Execution Time	
	DS1006	DS1006 Multicore
ds4121_init	87.79 $\mu$ s	109.03 $\mu$ s
ds4121_fifo_check	0.66 $\mu$ s	0.61 $\mu$ s
ds4121_fifo_reset	0.66 $\mu$ s	0.63 $\mu$ s
ds4121_fifo_start	0.66 $\mu$ s	0.63 $\mu$ s
ds4121_connection_check	0.82 $\mu$ s	0.66 $\mu$ s

### Functions for reading data from the ECU

The following execution times have been measured for the functions used to read data from the ECU:

Function	Execution Time		
	DS1005	DS1006	DS1006 Multicore
ds4121_p_int_read8	$(0.153 + n \cdot 0.200) \mu s^{1)}$	$(-0.522 + n \cdot 0.529) \mu s^{1)}$	$(0.146 + n \cdot 0.517) \mu s^{1)}$
ds4121_s_int_read8	$(0.214 + n \cdot 0.186) \mu s^{1)}$	$(-0.522 + n \cdot 0.529) \mu s^{1)}$	$(0.146 + n \cdot 0.517) \mu s^{1)}$
ds4121_le_p_int_read16, ds4121_le_p_int_read32, ds4121_le_s_int_read16, ds4121_le_s_int_read32	$(0.163 + n \cdot 0.190) \mu s,$ $(0.164 + n \cdot 0.387) \mu s,$ $(0.164 + n \cdot 0.387) \mu s,$ $(0.088 + n \cdot 1.062) \mu s^{1)}$	$(-0.515 + n \cdot 0.529) \mu s$ $(-1.041 + n \cdot 1.057) \mu s$ $(-1.042 + n \cdot 1.057) \mu s$ $(-2.088 + n \cdot 2.111) \mu s^{1)}$	$(0.146 + n \cdot 0.517) \mu s$ $(0.138 + n \cdot 1.046) \mu s$ $(0.139 + n \cdot 1.046) \mu s$ $(0.147 + n \cdot 2.075) \mu s^{1)}$
ds4121_be_p_int_read16, ds4121_be_p_int_read32, ds4121_be_s_int_read16, ds4121_be_s_int_read32	$(0.162 + n \cdot 0.190) \mu s,$ $(0.164 + n \cdot 0.387) \mu s,$ $(0.163 + n \cdot 0.387) \mu s,$ $(0.152 + n \cdot 0.749) \mu s^{1)}$	$(-0.515 + n \cdot 0.529) \mu s$ $(-1.042 + n \cdot 1.057) \mu s$ $(-1.041 + n \cdot 1.057) \mu s$ $(-2.094 + n \cdot 2.111) \mu s^{1)}$	$(0.146 + n \cdot 0.517) \mu s$ $(0.136 + n \cdot 1.046) \mu s$ $(0.138 + n \cdot 1.046) \mu s$ $(0.136 + n \cdot 2.092) \mu s^{1)}$
ds4121_le_p_fl_read32, ds4121_le_s_fl_read32, ds4121_be_p_fl_read32, ds4121_be_s_fl_read32	$(0.164 + n \cdot 0.387) \mu s$ $(0.126 + n \cdot 1.074) \mu s$ $(0.164 + n \cdot 0.387) \mu s$ $(0.188 + n \cdot 0.762) \mu s^{1)}$	$(-1.044 + n \cdot 1.059) \mu s$ $(-2.066 + n \cdot 2.089) \mu s$ $(-1.044 + n \cdot 1.059) \mu s$ $(-2.073 + n \cdot 2.089) \mu s^{1)}$	$(0.139 + n \cdot 1.046) \mu s$ $(0.145 + n \cdot 2.076) \mu s$ $(0.139 + n \cdot 1.046) \mu s$ $(0.138 + n \cdot 2.092) \mu s^{1)}$

<sup>1)</sup> n is the number of values to be read.

### Functions for transferring data to the ECU

The following execution times have been measured for the functions used to transfer data to the ECU:

Function	Execution Time		
	DS1005	DS1006	DS1006 Multicore
ds4121_p_int_write8	$(0.264 + n \cdot 0.141) \mu s^{1)}$	$(-0.085 + n \cdot 0.138) \mu s^{1)}$	$(0.509 + n \cdot 0.139) \mu s^{1)}$
ds4121_s_int_write8	$(0.262 + n \cdot 0.141) \mu s^{1)}$	$(-0.083 + n \cdot 0.136) \mu s^{1)}$	$(0.510 + n \cdot 0.139) \mu s^{1)}$
ds4121_le_p_int_write16, ds4121_le_p_int_write32, ds4121_le_s_int_write16, ds4121_le_s_int_write32	$(0.261 + n \cdot 0.141) \mu s$ $(0.114 + n \cdot 0.291) \mu s$ $(0.112 + n \cdot 0.291) \mu s$ $(-0.080 + n \cdot 0.583) \mu s^{1)}$	$(-0.083 + n \cdot 0.136) \mu s$ $(-0.248 + n \cdot 0.309) \mu s$ $(-0.260 + n \cdot 0.309) \mu s$ $(0.003 + n \cdot 0.606) \mu s^{1)}$	$(0.510 + n \cdot 0.139) \mu s$ $(0.352 + n \cdot 0.303) \mu s$ $(0.353 + n \cdot 0.303) \mu s$ $(0.057 + n \cdot 0.613) \mu s^{1)}$
ds4121_be_p_int_write16, ds4121_be_p_int_write32, ds4121_be_s_int_write16, ds4121_be_s_int_write32	$(0.261 + n \cdot 0.141) \mu s$ $(0.114 + n \cdot 0.291) \mu s$ $0.113 + n \cdot 0.291) \mu s$ $(-0.081 + n \cdot 0.583) \mu s$	$(-0.086 + n \cdot 0.135) \mu s$ $(-0.249 + n \cdot 0.309) \mu s$ $(-0.249 + n \cdot 0.309) \mu s$ $(0.002 + n \cdot 0.606) \mu s^{1)}$	$(0.513 + n \cdot 0.139) \mu s$ $(0.353 + n \cdot 0.303) \mu s$ $(0.352 + n \cdot 0.303) \mu s$ $(0.059 + n \cdot 0.613) \mu s^{1)}$



Function	Execution Time		
	DS1005	DS1006	DS1006 Multicore
ds4121_le_p_fl_write32, ds4121_le_s_fl_write32, ds4121_be_p_fl_write32, ds4121_be_s_fl_write32	(0.163 + n · 0.290) µs (-0.081 + n · 0.583) µs (0.161 + n · 0.290) µs (-0.080 + n · 0.583) µs <sup>1)</sup>	(-0.231 + n · 0.308) µs (-0.522 + n · 0.611) µs (-0.226 + n · 0.308) µs (-0.521 + n · 0.611) µs <sup>1)</sup>	(0.355 + n · 0.303) µs (0.058 + n · 0.613) µs (0.352 + n · 0.303) µs (0.074 + n · 0.613) µs <sup>1)</sup>

<sup>1)</sup> n is the number of values to be written.



**A**

abbreviations 8

**B**

big endian 8

**C**

Common Program Data folder 6  
 connection to the ECU 12  
 creating an ECU working directory 50

**D**

data layout in the DPMEM 9  
 data-typed read and write access 8  
 Documents folder 6  
 ds4121\_be\_p\_fl\_write32 32  
 ds4121\_be\_p\_int\_write16 31  
 ds4121\_be\_p\_int\_write32 31  
 ds4121\_be\_s\_fl\_write32 32  
 ds4121\_be\_s\_int\_write16 31  
 ds4121\_be\_s\_int\_write32 31  
 ds4121\_connection\_check 12  
 ds4121\_ecu\_version\_check 16  
 ds4121\_end\_of\_int\_set 19  
 ds4121\_fifo\_check 13  
 ds4121\_fifo\_reset 14  
 ds4121\_fifo\_start 15  
 ds4121\_init 11  
 ds4121\_le\_p\_fl\_write32 32  
 ds4121\_le\_p\_int\_write16 30  
 ds4121\_le\_p\_int\_write32 30  
 ds4121\_le\_s\_fl\_write32 32  
 ds4121\_le\_s\_int\_write16 30  
 ds4121\_le\_s\_int\_write32 30  
 ds4121\_p\_int\_read8 21  
 ds4121\_p\_int\_write8 27  
 ds4121\_s\_int\_read8 22  
 ds4121\_s\_int\_write8 29  
 ds4121\_subint\_init 18  
 dsecu\_alive 41  
 dsecu\_define\_sender 39  
 dsecu\_eosi\_poll 48  
 DSECU\_EOSI\_POLL 47  
 dsecu\_sender\_suspend 43  
 DSECU\_SINT\_INIT 37  
 dsecu\_sint\_send 46  
 DSECU\_SINT\_SEND 45  
 dsecu\_startup 40  
 dsECUframe.c 50  
 dsECUnew.bat 50  
 dssint\_subint\_enable 54  
 dssint\_wb\_acknowledge 56  
 dssint\_wb\_alive 60  
 dssint\_wb\_decode 55  
 dssint\_wb\_define\_receiver 52  
 dssint\_wb\_end\_of\_sint\_set 58  
 dssint\_wb\_startup 58  
 dssint\_wb\_subint\_disable 53

dssint\_wb\_subint\_reset 57  
 dssint\_wb\_suspend 61

**E**

ECU  
   connection 12  
   reading data from 21  
   writing data to 27  
 ECU-specific settings 50

**F**

floating point 8

**I**

initializing subinterrupts 37  
 integer 8

**L**

little endian 8  
 Local Program Data folder 6

**P**

packed data 8  
   data arrangement in DPMEM 9

**R**

reading data from the ECU 21  
 real-time library 5  
 RTLib 5

**S**

scattered data 8  
   data arrangement in DPMEM 9  
 subinterrupt  
   handling 45  
 subinterrupt handling  
   word based 51  
 subinterrupts 35  
   initializing 37  
   triggering 45  
 systems functions 11

**T**

triggering  
   subinterrupts 45

**U**

using subinterrupts 35

**W**

word based subinterrupt handling 51  
 working directory  
   ECU 50  
 writing data to the ECU 27

