

DS4121 ECU Interface Board

Features

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.

© 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
Introduction to the Features of the DS4121	7
Board Architecture.....	7
Feature Overview.....	9
Board Interfaces.....	9
ECU Interface Unit	11
Connection of the DS4121 to the ECU.....	11
DS4121 Working Modes.....	13
Specifying DPMEM Addresses Seen from the ECU.....	15
Supported ECU Data Type Formats.....	18
Interrupt Handling	21
Interrupt Handling.....	21
ECU Software Porting Kit	23
Working with the ECU Software Porting Kit.....	23
Word-Based Subinterrupt Handling	25
Basics on Word-Based Subinterrupt Handling.....	25
Basics on the Alive Mechanism.....	26
Basics on Subinterrupt Handling.....	29
Index	33

About This Document

Content This document provides feature-oriented access to the information you need to implement the functions of the DS4121.

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.

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.

Introduction to the Features of the DS4121

Where to go from here

Information in this section

[Board Architecture..... 7](#)

The DS4121 ECU Interface Board is designed to provide the link between your prototype or production type electronic control unit (ECU) and a PHS-bus-based system. The DS4121 uses bidirectional high-speed LVDS (Low Voltage Differential Signaling) connections with 250 Mbit/s to connect the two systems.

[Feature Overview..... 9](#)

Provides an overview of the features of the DS4121.

[Board Interfaces..... 9](#)

The DS4121 has interfaces for connection to a PHS-bus-based system and external devices.

Information in other sections

[DS4121 Data Sheet \(PHS Bus System Hardware Reference !\[\]\(51514032c8ca341817228f39f1307b05_img.jpg\)\)](#)

Board Architecture

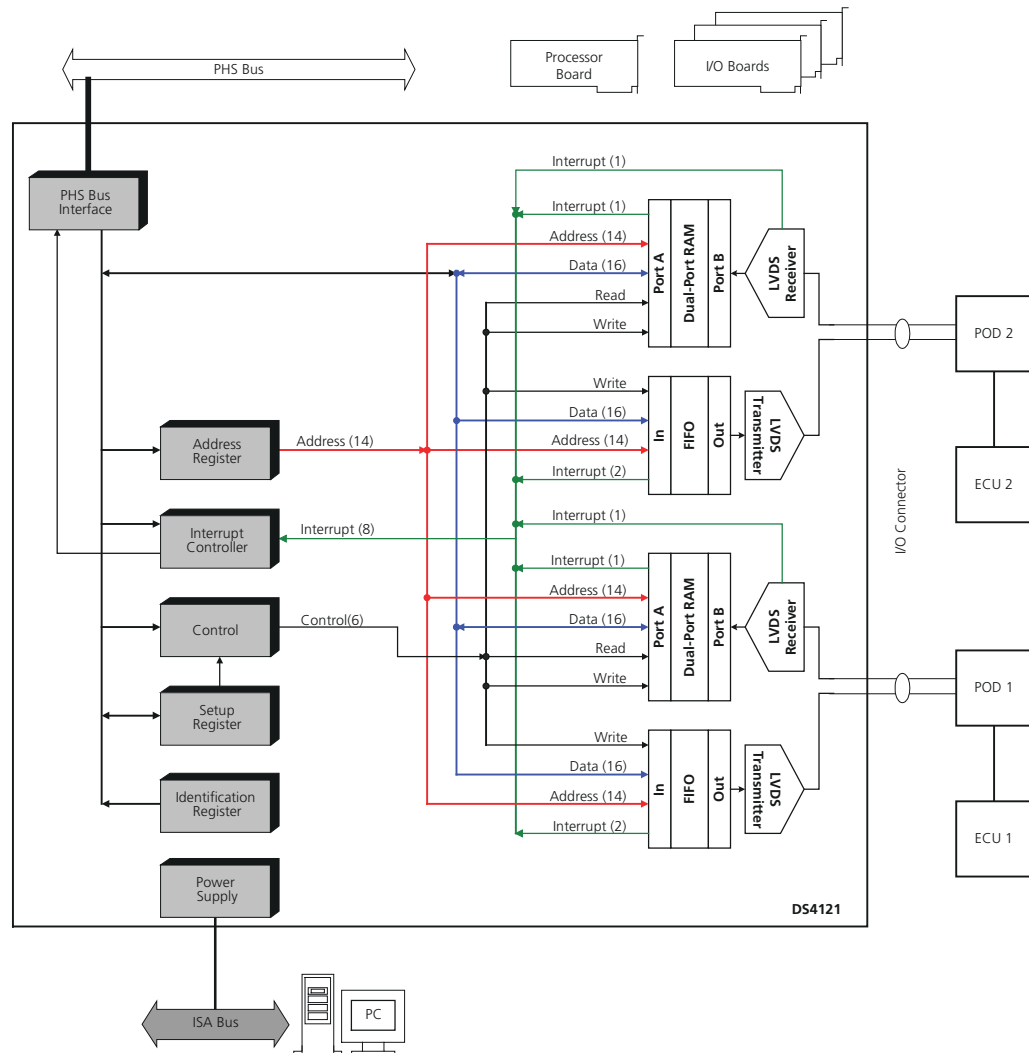
Introduction

The DS4121 ECU Interface Board is designed to provide the link between your prototype or production type electronic control unit (ECU) and a PHS-bus-based system. The DS4121 uses bidirectional high-speed LVDS (Low Voltage Differential Signaling) connections with 250 Mbit/s to connect the two systems.

DS4121-FX The DS4121-FX is similar to the DS4121. In contrast to the DS4121 it is connected to the ECU via fiber-optic cable.

Functional units

The following illustration shows the functional units of the DS4121.



Related topics

References

Board Interfaces.....	9
Feature Overview.....	9

Feature Overview

Introduction

The DS4121 ECU Interface Board lets you establish communication between an electronic control unit (ECU) and a dSPACE real-time system in combination with a custom-designed plug-on device (POD) that adapts the ECU signals to the DS4121 interface.

As a result, you can integrate the target ECU into the development of control algorithms performed on your dSPACE real-time system by using the DS4121 ECU Interface Board.

The DS4121 uses bidirectional high-speed LVDS (Low Voltage Differential Signaling) connections. You can connect up to two ECUs to the board.

C access functions ECUs differ from customer to customer regarding access mechanisms, memory layout, timings and pinout. Nevertheless, there is a common set of C access functions that performs the communication with the processor board.

ECU interface

Providing the communication to your prototype or your production type ECU. Refer to [ECU Interface Unit](#) on page 11.

Interrupt handling

Providing up to 8 slave interrupts. Refer to [Interrupt Handling](#) on page 21.

ECU software porting kit

Providing a software porting kit that allows the use of up to 16 subinterrupts. Refer to [ECU Software Porting Kit](#) on page 23.

Word-based subinterrupt handling

Providing subinterrupt handling over the DPMEM. Refer to [Word-Based Subinterrupt Handling](#) on page 25.

Related topics

References

[Board Interfaces](#)..... 9

Board Interfaces

Introduction

The DS4121 has interfaces for connection to a PHS-bus-based system and external devices.

Integration into a PHS-bus-based system

To be used, the DS4121 must be integrated into a PHS-bus-based system. While the DS4121 carries out the communication to your prototype, the processor board takes over the calculation of the real-time model. That is, applications using DS4121 ECU interface features are implemented on the processor board.

Communication between the processor board and I/O boards is performed via the peripheral high-speed bus: That is the PHS bus for a connection to a dSPACE processor board.

Partitioning the PHS bus with the DS802 With the DS802 PHS Link Board you can spatially partition the PHS bus by arranging the I/O boards in several expansion boxes.

The DS802 can be used in combination with many types of available dSPACE I/O boards. However, some I/O boards and some functionalities of specific I/O boards are not supported.

The I/O board support depends on the dSPACE software release which you use. For a list of supported I/O boards, refer to [DS802 Data Sheet \(PHS Bus System Hardware Reference !\[\]\(a870788d6ed9b8fd294b7654a8c8526b_img.jpg\)](#)).

Connection to external devices

To connect the communication channels 1 and 2 to the respective ECU via customdesigned PODs, use interface connectors P1 and P2. Refer to [Connection to the ECU \(PHS Bus System Hardware Reference !\[\]\(3211b5d1d968fc1665909b34f9f16010_img.jpg\)](#)).

Related topics**References**

[Feature Overview.....9](#)

ECU Interface Unit

Where to go from here

Information in this section

Connection of the DS4121 to the ECU.....	11
Provides general information on the ECU interface.	
DS4121 Working Modes.....	13
Explains the possibilities to work with the ECU interface.	
Specifying DPMEM Addresses Seen from the ECU.....	15
Shows how to specify DPMEM addresses with or without start address.	
Supported ECU Data Type Formats.....	18
Provides information on the supported ECU data types.	

Connection of the DS4121 to the ECU

Introduction

In bypass-based prototyping, existing ECUs are optimized or partially revised to obtain new control algorithms. The ECU executes all the control functions that remain unchanged, while the new algorithms are calculated by the dSPACE hardware. The results are then transmitted back to the original ECU.

ECU interface unit of the DS4121

The ECU interface unit of the DS4121 has two ECU interfaces. Each interface provides a 16K · 16bit dual-port RAM (DPMEM) for ECU bypassing. You have to specify DPMEM addresses as *ECU addresses*. For more information, refer to [Specifying DPMEM Addresses Seen from the ECU](#) on page 15.

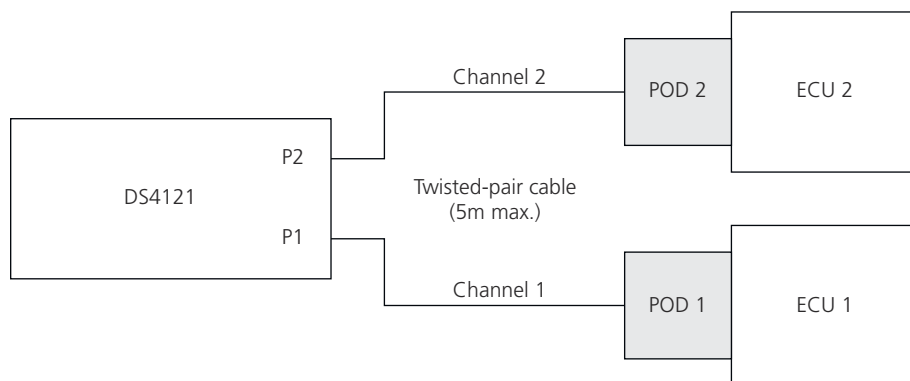
The communication between the ECU and the DS4121 can be controlled by (sub-)interrupts. For more information, refer to [Interrupt Handling](#) on page 21.

ECU connection to the DS4121

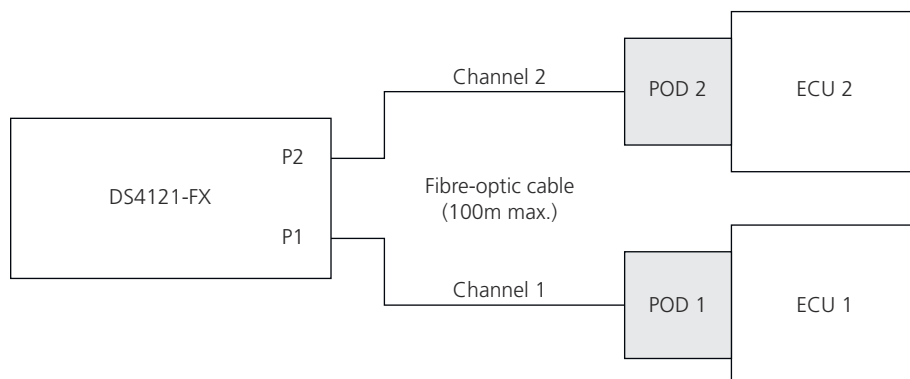
Plug-on device (POD) The ECU is connected to the DS4121 or DS4121-FX via a *plug-on device* (POD).

In typical applications, a custom-specific plug-on device (POD) is necessary. The POD provides the physical connection between the DS4121 or DS4121-FX and the ECU and performs signal conditioning, for example.

Connection between the DS4121 and the ECU The following illustration shows the connection between the DS4121 and two ECUs:



Connection between the DS4121-FX and the ECU The following illustration shows the connection between the DS4121-FX and two ECUs:



ECU configuration file

The hardware-specific settings for your ECU are defined in the *ECU configuration file* provided by dSPACE. The ECU configuration file specifies the ECU used with the dSPACE POD and contains the ECU offset address, the POD offset address, and other necessary parameters.

For further information on the hardware and the complete I/O mapping of the board, refer to [DS4121 Data Sheet \(PHS Bus System Hardware Reference \[1\]\)](#).

Related topics

References

[DS4121 Data Sheet \(PHS Bus System Hardware Reference !\[\]\(bd1a142de767a21e5362c595f844a4ff_img.jpg\)\)](#)
[ECU Interface Unit \(DS4121 RTI Reference !\[\]\(d4257ae6a3e163e6d467b3eb87960fa1_img.jpg\)\)](#)

DS4121 Working Modes

Introduction

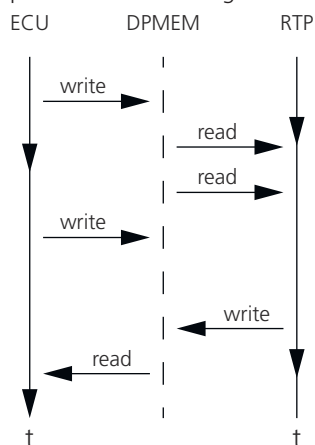
The ECU interface supports three working modes. In your ECU application, you determine the mode to be used.

Note

The working mode actually used depends on the usage of the ECU interface in your application. You cannot set the working mode using RTI blocks or RTLib functions.

Asynchronous mode

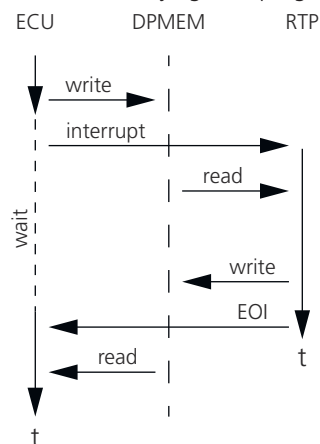
In this mode no synchronization between ECU and RTP is performed. The ECU does not generate interrupts and does not wait for status messages from the RTP. Since both units access the DPMEM independently, there is a danger of data inconsistencies or even data losses – that means, it is uncertain whether the values read from the DPMEM are new or old. In general, this mode is used for parameter monitoring and for the tuning of non time-critical parameters.



Synchronous mode

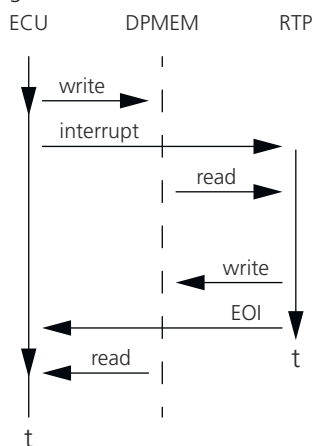
The synchronization between ECU and RTP is performed by an interrupt mechanism. For Simulink models, the interrupt (from ECU to RTP) triggers an interrupt-driven subsystem. In this subsystem, the read and write access to the DPMEM is performed: The ECU writes the value to the DPMEM, generates an interrupt and waits for the "End of interrupt" message (EOI) from the RTP. The

RTP reads this value, performs the necessary calculations, writes the calculated new value back to the DPMEM and generates the "End of interrupt" message. Following, the ECU will read the calculated new value. In general, this mode is used for modifying ECU programs.



Semi-synchronous mode

This – simulated bypassing – mode corresponds to the synchronous mode except, that the ECU does not wait for an "End of interrupt" message from the RTP. Data inconsistencies may occur if the ECU reads data from the DPMEM before the calculations of the RTP in the interrupt routine or the interrupt-driven subsystem are finished. To ensure correct data the ECU should access the DPMEM not before the "End of interrupt" message from the RTP has been generated or after the RTP has finished its calculations.



Related topics

Basics

[ECU Interface Unit..... 11](#)

Specifying DPMEM Addresses Seen from the ECU

Introduction

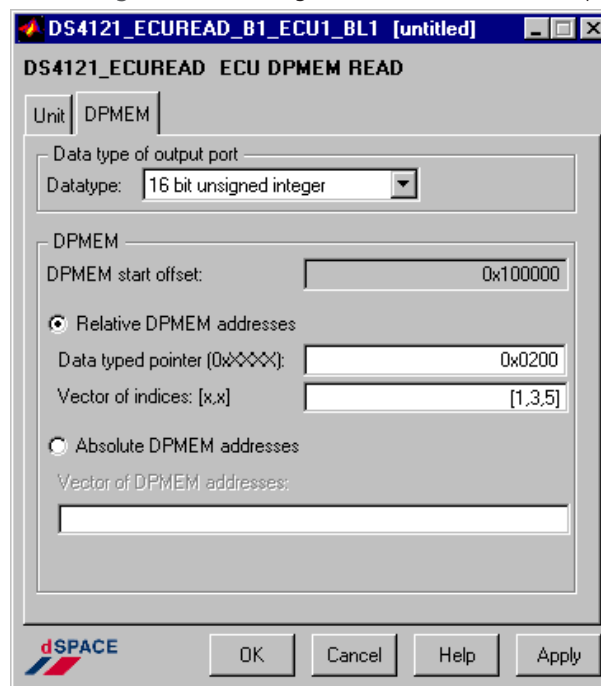
The DS4121 dual-port RAM (DPMEM) is mapped into the memory of the ECU. dSPACE's RTI dialogs allow you to specify DPMEM addresses as *ECU addresses*, i.e., DPMEM addresses as seen from the ECU. You can use absolute or relative addresses with or without start offset.

The conversion algorithm depends on the ECU offset address and other parameters specified in the ECU configuration file. The ECU address offset specifies the start address of the DPMEM address range seen from the ECU side.

Relative addressing with start address

This mode adds the relative data addresses to the DPMEM start address.

RTI setting The following illustration shows an example for the RTI settings:



Corresponding ECU code The following example shows the ECU code corresponding to the above RTI settings:

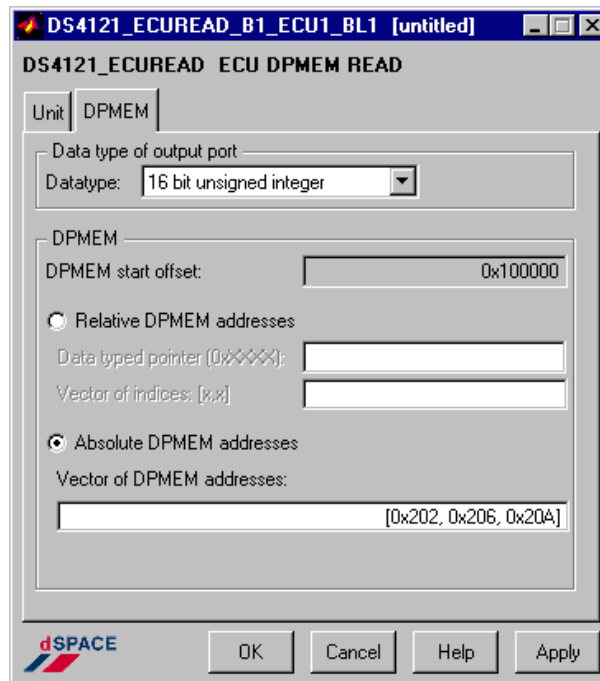
```
#define DATA_BASE      0x00100000  /* Corresponds to a setting in the ECU */
                               /* configuration file; ECU specific */
#define DATA_INPUT_BLOCK 0x00000200 /* Corresponds to the GUI setting in */
                               /* "data typed pointer" field */

{
    volatile unsigned short *input =
(volatile unsigned short *) (DATA_BASE + DATA_INPUT_BLOCK);
    unsigned short a,b,c;
    a = input[1]; /* Read first value from dSPACE system */
    b = input[3]; /* Read 2nd value from dSPACE system */
    c = input[5]; /* Read 3rd value from dSPACE system */
}
```

Absolute addressing with start address

This mode adds the absolute data addresses to the DPMEM start address.

RTI setting The following illustration shows an example for the RTI settings:



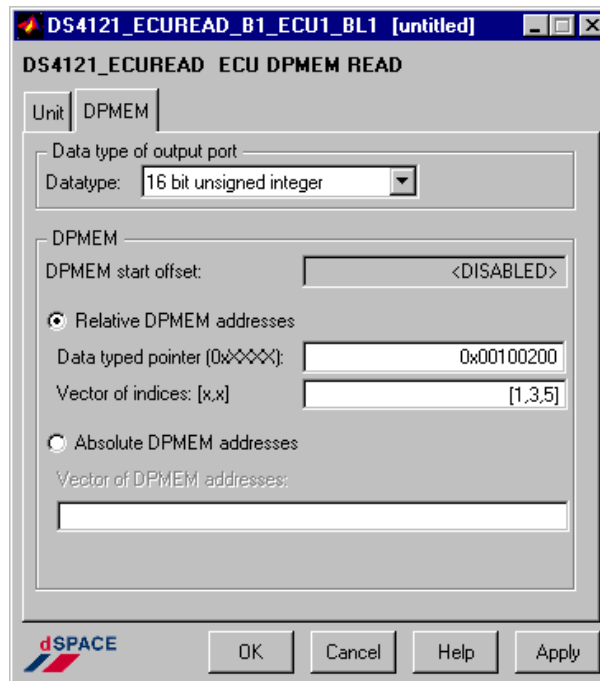
Corresponding ECU code The following example shows the ECU code corresponding to the above RTI settings:

```
#define DATA_BASE    0x00100000 /* Corresponds to a setting in the ECU */
                          /* configuration file; ECU specific */
#define input(A)      (*(volatile unsigned short *) (DATA_BASE+A))
{
  unsigned short a,b,c;
  a = input(0x202); /* Read first value from dSPACE system */
  b = input(0x206); /* Read 2nd value from dSPACE system */
  c = input(0x20A); /* Read 3rd value from dSPACE system */
}
```


Relative addressing without start address

This mode uses relative data addresses to access the DPMEM.

RTI setting The following illustration shows an example for the RTI settings:



Corresponding ECU code The following example shows the ECU code corresponding to the above RTI settings:

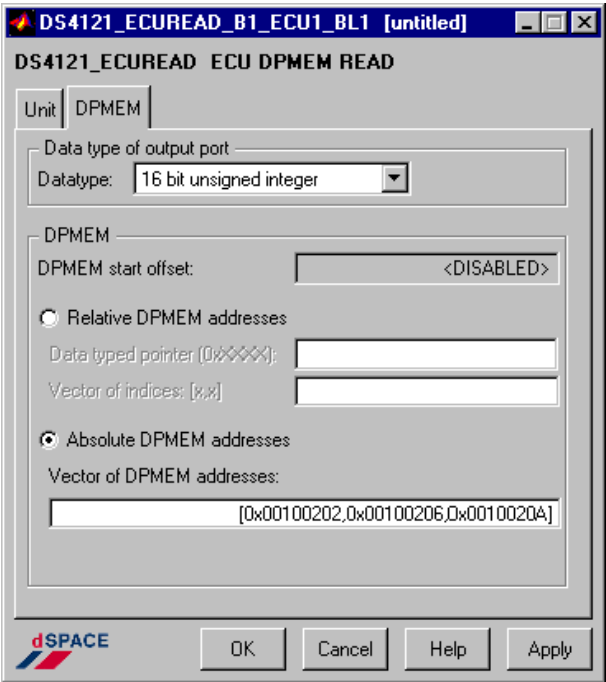
```
#define DATA_INPUT_BLOCK 0x00100200 /* Corresponds to the GUI setting in */
/* "data typed pointer" field */

{
    volatile unsigned short *input = (volatile unsigned short *) (DATA_INPUT_BLOCK);
    unsigned short a,b,c;
    a = input[1]; /* Read first value from dSPACE system */
    b = input[3]; /* Read 2nd value from dSPACE system */
    c = input[5]; /* Read 3rd value from dSPACE system */
}
```

Absolute addressing without start address

This mode uses absolute data addresses to access the DPMEM.

RTI setting The following illustration shows an example for the RTI settings:



Corresponding ECU code The following example shows the ECU code corresponding to the above RTI settings:

```
{
unsigned short a,b,c;
a = *((volatile unsigned short *) 0x0010202); /* Read first value from dSPACE system */
b = *((volatile unsigned short *) 0x0010206); /* Read 2nd value from dSPACE system */
c = *((volatile unsigned short *) 0x001020A); /* Read 3rd value from dSPACE system */
}
```

Related topics

Basics	
Connection of the DS4121 to the ECU.....	11
Supported ECU Data Type Formats.....	18

Supported ECU Data Type Formats

Overview

ECUs use different data formats and different data bus formats. dSPACE’s RTI supports the following data types:

- 8 bit signed integer
- 8 bit unsigned integer
- 16 bit signed integer
- 16 bit unsigned integer
- 32 bit signed integer
- 32 bit unsigned integer
- Single float 32 bit (IEEE Std. 754)

This means, that the dSPACE software

- reads bit patterns from the DPMEM and provides the data in the given format for processing in Simulink or
- provides data of a Simulink variable in the given format to be written to the DPMEM.

Note

For some data formats, two or more subsequent addresses are used.

- For 16-bit data, only even addresses have to be used.
- For 32-bit data the addresses have to be multiples of 4. This refers to the 8-bit view of the DPMEM from the ECU. Consider this, when entering the read or write address for absolute addressing. Relative addressing takes this into account automatically.

RTI provides a check on overlapping DPMEM addresses. If you use hand-coded models, you have to ensure correct addresses yourself.

To connect other Simulink blocks to the RTI blocks managing the DPMEM access, you have to use a Data Conversion block to adapt the data format.

Related topics

Basics

[Specifying DPMEM Addresses Seen from the ECU..... 15](#)

References

[DS4121_ECUREAD_Bx_CHy_BLz \(DS4121 RTI Reference !\[\]\(b792654f2cef9719eabeb6c5be00811e_img.jpg\)](#))

[DS4121_ECUWRITE_Bx_CHy_BLz \(DS4121 RTI Reference !\[\]\(7da9a585536d56657fa124d7eaae44e7_img.jpg\)](#))

Interrupt Handling

Interrupt Handling

Interrupt sources

The DS4121 provides eight interrupt sources:

Interrupt Source	Description
INT0	Interrupt from the DPMEM, channel 1. See below for more information.
INT1	Connection failed for channel 1
INT2	FIFO almost full for channel 1 (less than 64 words free)
INT3	FIFO full for channel 1
INT4	Interrupt from the DPMEM, channel 2. See below for more information.
INT5	Connection failed for channel 2
INT6	FIFO almost full for channel 2 (less than 64 words free)
INT7	FIFO full for channel 2

For further information on the hardware, refer to [DS4121 Data Sheet \(PHS Bus System Hardware Reference !\[\]\(0aff635c4179ba9e710b00f4b01d3b20_img.jpg\)](#)).

ECU to RTP interrupts

The interrupts INT0 or INT4 are used to synchronize the data transfer. These interrupts are called *ECU-to-RTP interrupts* and are handled via the DPMEM:

- To generate an interrupt from the ECU to the RTP, the ECU writes a value to a certain DPMEM address.
- The RTP clears an interrupt by reading this value.

Subinterrupts on the ECU side

On the ECU side, each ECU-to-RTP interrupt can be split into 16 subinterrupts via the ECU Software Porting Kit (refer to [ECU Software Porting Kit](#) on page 23).

Note

If you do not use dSPACE's ECU Software Porting Kit, you have to program an equivalent subinterrupt handling yourself to ensure proper communication between ECU and RTP.

Subinterrupts on the RTP side

Interrupt-driven subsystems in RTI You can use interrupts or subinterrupts to trigger interrupt-driven subsystems of your Simulink model. When the task in this system has finished, the interrupt handling creates automatically an "End of interrupt" (EOI) message to indicate the state for other units, an ECU, for example.

Hand-coded models If you use hand-coded models, you can use the functions provided by dSPACE's Word-Based Subinterrupt Handling to program the subinterrupt handling on the RTP side yourself (refer to [Word-Based Subinterrupt Handling](#) on page 25).

Related topics

Basics

[ECU Software Porting Kit](#)..... 23

References

[DS4121_ECUINT_Bx_CHy_I0, DS4121_ECUINT_Bx_CHy_Siz \(DS4121 RTI Reference !\[\]\(d5d7044e5caf6907399af2dced8d6ff8_img.jpg\)](#))

ECU Software Porting Kit

Working with the ECU Software Porting Kit

Basics

The ECU Software Porting Kit allows you to use up to 16 subinterrupts (0 ... 15) from the ECU to the DS4121. You can use the subinterrupts without detailed knowledge of the subinterrupt functionality.

The settings needed for your specific ECU must be specified in the `dsECUframe.c` file. This file allows you to define, among others, the type of the ECU interface board and the DPMEM addresses used for (sub-)interrupt handling. All other required information are included in this file.

Tip

You have to modify existing ECU code only if you want to use subinterrupts in your application.

For further information, refer to [ECU Software Porting Kit \(DS4121 RTLib Reference !\[\]\(2b376d1a92330ab09dad2665d2f89bf5_img.jpg\)](#)).

Using the ECU Software Porting Kit

If you want to use the ECU Software Porting Kit to set subinterrupts you have to perform the following steps:

- Create the working directory for the ECU application and copy `dsECUframe.c` to this directory. For details, refer to [dsECUnew.bat \(DS4121 RTLib Reference !\[\]\(029651ce9ee64da8525b17c64e266edc_img.jpg\)](#)).
- For existing ECU projects copy this file to the working directory.
- Specify the settings for your specific ECU in the file `dsECUframe.c`. For details refer to [Defines to be Specified in the dsECUframe.c File \(DS4121 RTLib Reference !\[\]\(05d3bfcaecedf25939aadd260bd34af7_img.jpg\)](#)).
- Include `dsECUframe.c` into your application.
- You can now perform subinterrupt handling in your application code:
 - `DSECU_SINT_INIT` calls `dsecu_define_sender` to initialize the interrupt sender.

- DSSINT_SINT_SEND calls `dsecu_sint_send` to trigger a subinterrupt and set the end-of-subinterrupt flag.
- DSECU_EOSI_POLL polls the end-of-subinterrupt flag.

Example

The following example demonstrates how to use the ECU Software Porting Kit:

```
#include <stdlib.h>
#include <dsECUframe.c>
void main(void)
{
    unsigned int sint_number;
    ();
    while(1)
    {
        static int first_alive_flag = 1;    /* used for revive */
        if (!first_alive_flag)
        {
            (&dssint_sender); /* first alive only one time to revive ECU */
            first_alive_flag = 1;
        }
        sint_number = rand();    /* random subinterrupt */
        write_bypass_data(sint_number);
        (sint_number);
        if ((sint_number) == DSECU_BYPASS_DATA_VALID)
        {
            read_bypass_data(sint_number);
        }
    }
}
```

Related topics

References

[dsecu_define_sender \(DS4121 RTLib Reference !\[\]\(17acf1afa8cdf0b67c53d4865a5ed469_img.jpg\)\)](#)
[DSECU_EOSI_POLL \(DS4121 RTLib Reference !\[\]\(ece8cabb5adcd402275b8866019cc3b8_img.jpg\)\)](#)
[DSECU_SINT_INIT \(DS4121 RTLib Reference !\[\]\(4fe6c1f6e7bbe5a2699a4abd6267bb58_img.jpg\)\)](#)
[dsecu_sint_send \(DS4121 RTLib Reference !\[\]\(70a50cebc68af4280759ff1f65916f6e_img.jpg\)\)](#)
[dsecu_startup \(DS4121 RTLib Reference !\[\]\(a864435f938b4616d4c31924501fac76_img.jpg\)\)](#)

Word-Based Subinterrupt Handling

Introduction Information on how to handle the subinterrupts generated by an ECU.

Where to go from here Information in this section

[Basics on Word-Based Subinterrupt Handling.....](#) 25
The Word-Based Subinterrupt module is designed to implement the subinterrupt features on the real-time processor (RTP) side.

[Basics on the Alive Mechanism.....](#) 26
Correct subinterrupt handling requires that the corresponding partner system is running ("alive").

[Basics on Subinterrupt Handling.....](#) 29
Provides information on the subinterrupt handling.

Basics on Word-Based Subinterrupt Handling

Introduction The Word-Based Subinterrupt module is designed to implement the subinterrupt features on the real-time processor (RTP) side, that is your dSPACE real-time system, for example, a DS1007-based modular system or MicroAutoBox II.

Note

To implement a subinterrupt sender on the ECU without intervention to the interrupt state of the ECU, use the [ECU Software Porting Kit \(DS4121 RTLib Reference !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\)](#)).

Features of the Word-Based Subinterrupt Handling module

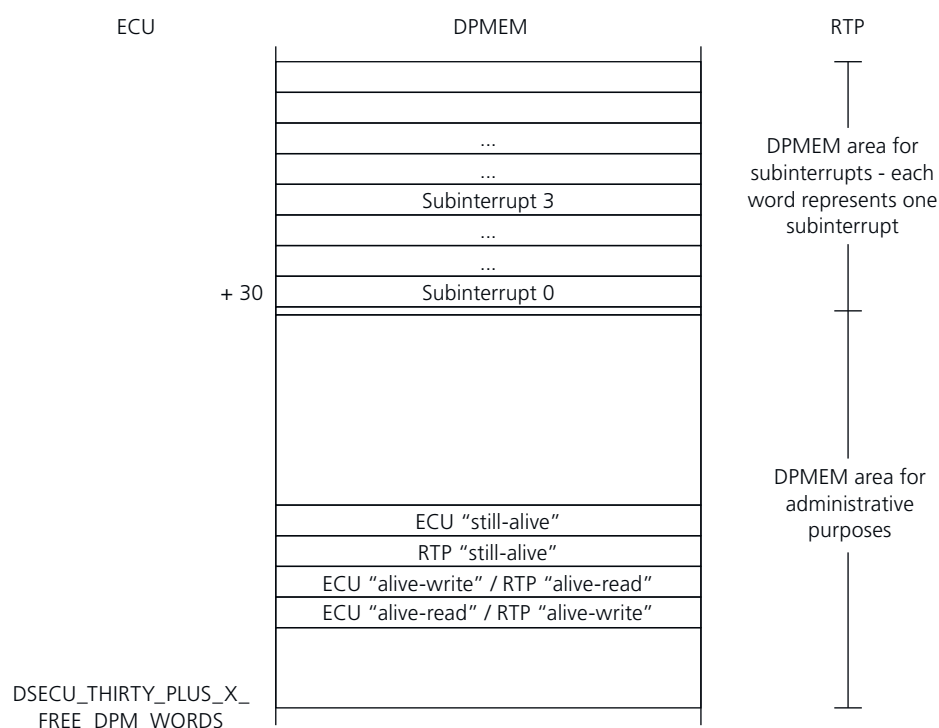
The Word-Based Subinterrupt Handling module provides the following features:

- It lets you trigger and handle multiple subinterrupts using a single hardware interrupt line.
- It lets you specify multiple subinterrupts as *pending* at the receiver.
- It lets you transmit and dispatch interrupts between an ECU and a dSPACE RCP system.
- It lets you define interrupt senders/receivers to transmit subinterrupts.

DPMEM usage

The Word-Based Subinterrupt Handling module requires $30 + x$ (with x = number of subinterrupts to be used) successive DPMEM addresses. The first 30 words are used for the alive mechanism, version information and other administrative purposes.

The following illustration shows the usage of the reserved DPMEM areas and the location of the words used for the alive mechanism and the subinterrupt handling:



Basics on the Alive Mechanism

Introduction

Correct subinterrupt handling requires that the corresponding partner system is running ("alive").

Alive state

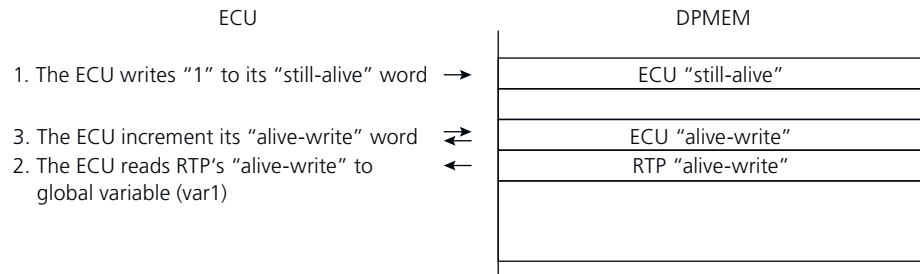
The partner system is supposed to be "alive" if the contents of the own "alive-read" word is changed. The own "alive-read" is equal to the partner's "alive-write" word. After the alive state is recognized, the "still-alive" word of the partner is evaluated.

Startup function

The alive startup function

- sets the own "still-alive" word
- reads the partner's "alive-write" word to initialize its compare value which is used to recognize changes of this word
- increments its own "alive-write" word for the partner

The following illustration shows the activities from the ECU side (the RTP uses its DPMEM addresses to perform corresponding actions):

**Alive function**

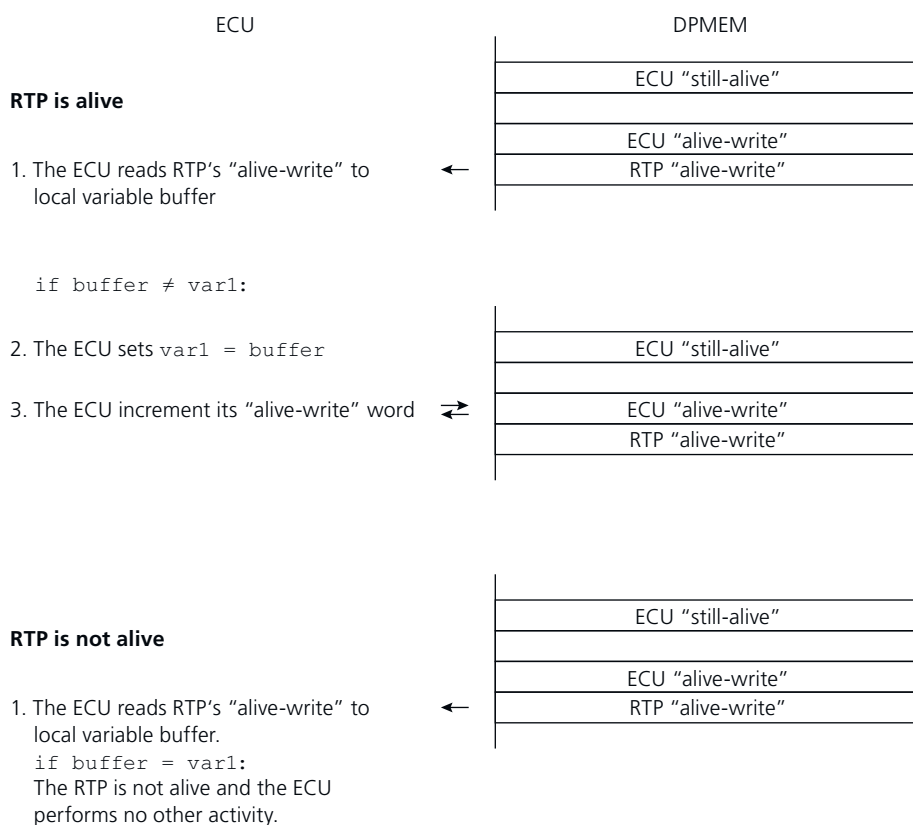
The alive function has to be called repetitively to compare the partner's "alive-write" word with the internal compare value. If the value has changed, the partner is recognized as alive. Only in this case, the own "alive-write" word is incremented again.

Note

This must not be done every time to keep the FIFO of the DS4121 empty.

In addition to this, the internal compare value is updated to the new value. After the partner is recognized as alive, only the "still-alive" word is checked.

The following illustration shows the activities from the ECU side. The RTP uses its DPMEM addresses to perform analog actions.



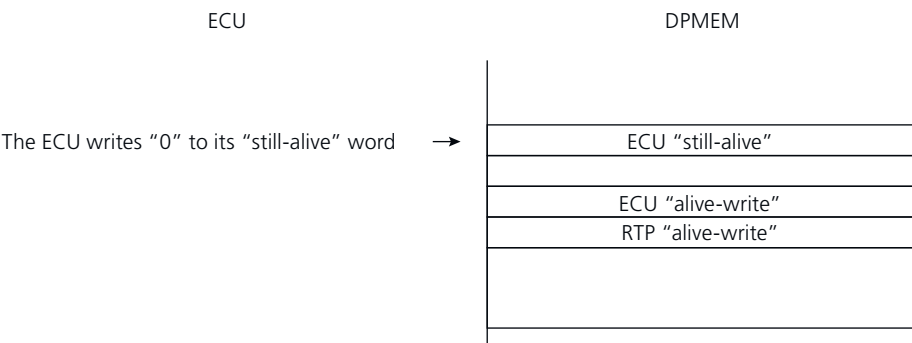
Suspend function

If the ECU or the dSPACE RCP system intends to go to "not alive", it must

- call the suspend function which clears the "still-alive" word and
- must not call the alive function any longer.


The alive function of the other side notices that the "still-alive" word is cleared and checks the partner's "alive-write" word for changes. This allows the system to recognize when the partner goes to the alive state again.

To revive a system you have to call the startup function once before the alive function is called repetitively again.



For detailed information on the functions, refer to [Word-Based Subinterrupt Handling \(DS4121 RTLib Reference !\[\]\(bd1a142de767a21e5362c595f844a4ff_img.jpg\)](#)).

Related topics

Basics	
ECU Software Porting Kit.....	23
References	
Word-Based Subinterrupt Handling (DS4121 RTLib Reference 	

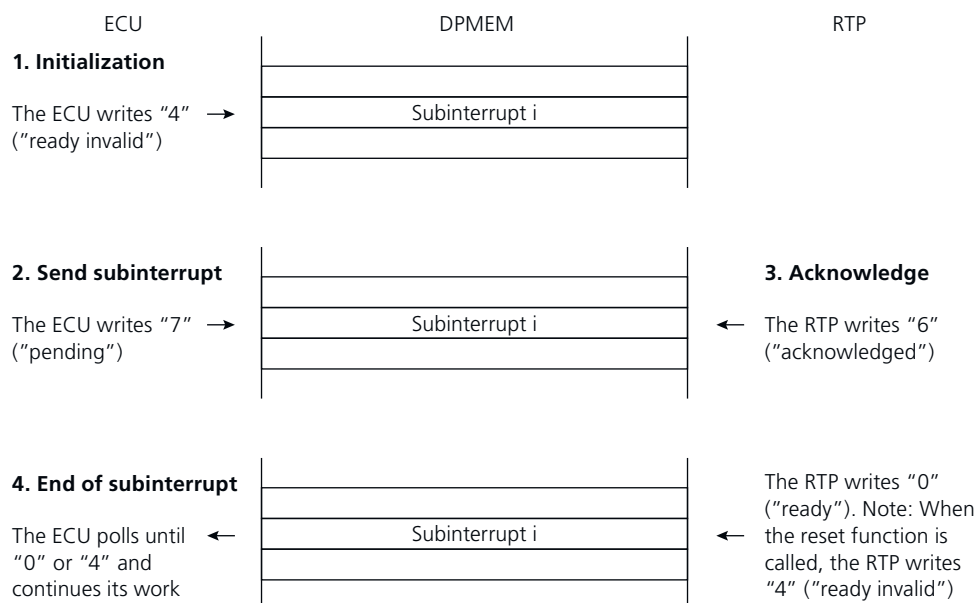
Basics on Subinterrupt Handling

Introduction

To implement word-based subinterrupt handling, you must consider the following working steps.

Subinterrupt handling overview

The following illustration shows the subinterrupt handling – for example, synchronous bypassing – in relation to the DPMEM addresses used:

**Initialization**

Subinterrupt sender and receiver initialize the subinterrupt words – for the defined number of subinterrupts – in the DPMEM with the value "ready invalid" (= 4).

Pending subinterrupt

If the receiver system is alive, the sender's send function sets the subinterrupt state to "pending" (= 7) and triggers the hardware interrupt.

Acknowledge



The receiver's interrupt handling performs the following steps:

- It calls the acknowledge function to acknowledge the hardware interrupt by reading the DPMEM acknowledge address.
- It sets all pending subinterrupts to the state "acknowledged" (= 6).

Decoding

The information about the acknowledged subinterrupts is passed to the *decode function* via an internal data structure. The decode function must be called repetitively to return every time the number of the subinterrupt that was acknowledged before.

If it returns -1 there is no more subinterrupt to handle.

End of subinterrupt	After the subinterrupt task is finished, the <i>end-of-subinterrupt (EOSI) function</i> has to be called. It sets the subinterrupt state to "ready" (= 0). This is the signal for the sender that the data can be read. The function that polls for the end-of-subinterrupt returns the "valid" state.
Reset	If the receiver calls the reset function for a subinterrupt, its state is set to "ready invalid" (= 4) again as if it was pending before. The sender's poll function returns the "invalid" state and the program can react appropriately.
Related topics	References <div>ECU Software Porting Kit (DS4121 RTLib Reference ) Word-Based Subinterrupt Handling (DS4121 RTLib Reference )</div>

A

- absolute address 15
- address
 - absolute 15
 - relative 15
 - start offset 15
- asynchronous mode 13

B

- bypass-based prototyping 11

C

- Common Program Data folder 6

D

- data type formats 18
- Documents folder 6
- DPMEM addresses 15
- DS4121-FX 7
- DS802
 - partitioning PHS bus 10

E

- ECU 7
 - data types 18
 - DPMEM addresses 15
 - interface unit
 - characteristics 11
- ECU addresses
 - examples 15
- ECU configuration file 12
- ECU Software Porting Kit 23
 - example 24
 - working with the 23

I

- INT 21
- IRQ 21

L

- Local Program Data folder 6

P

- partitioning PHS bus with DS802 10
- plug-on device 11
- POD 11

R

- relative address 15
- ribbon cable 11

S

- semi-synchronous mode 14
- start offset 15

- subinterrupt handling 21
 - word-based 25
- synchronous mode 13

W

- word-based subinterrupt handling 25
- working modes 13

