DS4002 Timing and Digital I/O Board

RTLib Reference

Release 2021-A – May 2021

dSPACE

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 and the contents hereof are subject to change without notice.

Contents

About This Reference

Contents

This reference (Real-Time Library) gives detailed descriptions of the C functions needed to program a DS4002 Timing and Digital I/O Board. The C functions can be used to program RTI-specific Simulink S-functions, or to implement your real-time models manually using C programs.

Demo examples

There are examples for some features included in this documentation. You will find the relevant files after the installation of your dSPACE software in `<RCP_HIL_InstallationPath>\Demos\Ds100x\IOBoards\Ds4002`. Use ControlDesk to load and start the application on your processor board.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
⚠ DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
⚠ WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
⚠ CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.
?	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
📖	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

Special folders

Some software products use the following special folders:

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files

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

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

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

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

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

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

Examples

Example of Using Interrupts

Introduction

The following example demonstrates how to use the interrupt sources of the DS4002 in a hand-coded application.

Note

Do not use hand-coded interrupt handling within S-functions. If you use the `DS4002_INT_CLEAR` macro, the access on the interrupt status field can conflict with interrupt requests (i.e., an edge count interrupt) specified within the Simulink application.

Description

Channel 1 generates a simple square-wave signal with interrupt generation at each rising edge. Channel 2 generates a square-wave signal with interrupt generation at each rising and falling edge. Channels 3 … 8 are used in input mode, each capturing rising edges. All channels generate an interrupt when 20 edges have been detected. They are updated in an interrupt service routine every 1 ms.

You have to connect the channel 1, 2 or an external signal to channels 3 … 8.

```
#include "brtenv.h"                    /* basic real time environment */
#include "ds4002.h"                    /* DS4002 constants and macros */
/***********************************************************************
  global variables
***********************************************************************/
dsfloat freq1    = 1000;                            /* initial values */
dsfloat freq2    = 1000;
```

```
long count1     = 0;
long count2     = 0;
long count3     = 0;
long count4     = 0;
long count5     = 0;
long count6     = 0;
long count7     = 0;
long count8     = 0;
long sum_count  = 0;
/*********************************************************************
  timer 1 interrupt service routine
**********************************************************************/
void isr_t1()
{
  RTLIB_SRT_ISR_BEGIN();                          /* overload check */
  RTLIB_INT_DISABLE();
  /* update channel 1 */
  ds4002_update_state(DS4002_1_BASE, 1, 0,
     DS4002_DELAY(0.5/freq1),                  /* after half period */
     DS4002_HIGH,                                /* set output high */
     DS4002_CONTINUE,                   /* continue with next state */
     0);                           /* no loop counter or jump value */
  ds4002_update_state(DS4002_1_BASE, 1, 1,
     DS4002_state(0.5/freq1),                  /* after half period */
     DS4002_LOW,                                  /* set output low */
     DS4002_GOTO,              /* goto entry point (= first state) */
     0);                           /* no loop counter or jump value */
  DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_NEWDATA, 1); /* advance
                            swinging buffer for use with next delay */
  /* update channel 2 */
  ds4002_update_state(DS4002_1_BASE, 2, 0,
     DS4002_DELAY(0.5/freq2),                  /* after half period */
     DS4002_HIGH,                                /* set output high */
     DS4002_CONTINUE,                   /* continue with next state */
     0);                           /* no loop counter or jump value */
  ds4002_update_state(DS4002_1_BASE, 2, 1,
     DS4002_DELAY(0.5/freq2),                  /* after half period */
     DS4002_LOW,                                  /* set output low */
     DS4002_GOTO,              /* goto entry point (= first state) */
     0);                           /* no loop counter or jump value */
  DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_NEWDATA, 2); /* advance
                            swinging buffer for use with next delay */
  RTLIB_INT_ENABLE();
  RTLIB_SRT_ISR_END();                            /* overload check */
}
/*********************************************************************
  channel 1 interrupt service routine
**********************************************************************/
void channel1_intserv()
{
  count1 += 1;
}
/*********************************************************************
  channel 2 interrupt service routine
**********************************************************************/
void channel2_intserv()
{
  count2 += 1;
}
```

```
/**********************************************************************/
void single_channel (long channel, long *count, long mask)
{
  long time[20];
  long state[20];
  long c;
  (*count)++;
  c = 20;
  ds4002_read_contiguous(DS4002_1_BASE, channel, &c, time, state);
  DS4002_INT_CLEAR(DS4002_1_BASE, mask);
}
/**********************************************************************/
void ilen_intserv()
{
  sum_count ++;
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x01)
    single_channel(1, &count1, 0x01);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x02)
    single_channel(2, &count2, 0x02);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x04)
    single_channel(3, &count3, 0x04);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x08)
    single_channel(4, &count4, 0x08);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x10)
    single_channel(5, &count5, 0x10);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x20)
    single_channel(6, &count6, 0x20);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x40)
    single_channel(7, &count7, 0x40);
  if (DS4002_INT_STATUS(DS4002_1_BASE ) & 0x80)
    single_channel(8, &count8, 0x80);
}
/*************************************************************************
  main
*************************************************************************/
void main()
{
  init();                          /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);  /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  install_phs_int_vector(        /* initialize interrupt controllers */
     DS4002_1_BASE,           /* board base address */
     0,                      /* slave ICU input
                                0 = ILEN interrupt in input mode
                                (check INT register for channel numbers) */
     ilen_intserv );                  /* address of service routine */
  install_phs_int_vector(       /* initialize interrupt controllers   */
     DS4002_1_BASE,          /* board base address */
     1,                     /* slave ICU input
                               1 = channel 1 in output mode         */
     channel1_intserv );              /* address of service routine */
  install_phs_int_vector(       /* initialize interrupt controllers */
     DS4002_1_BASE,          /* board base address */
     2,                     /* slave ICU input
                             2 = channel 2 in output mode           */
     channel2_intserv );              /* address of service routine */
  /* ch1: ftod */
  ds4002_output_init();                /* prepare program variables */
  ds4002_define_entry();             /* entry point = program start */
  ds4002_define_state(
     DS4002_DELAY(0.5/freq1),                  /* after half period */
     DS4002_HIGH,                                /* set output high */
     DS4002_INTERRUPT,                            /* host interrupt */
     DS4002_CONTINUE,                   /* continue with next state */
     0);                           /* no loop counter or jump value */
  ds4002_define_state(
     DS4002_DELAY(0.5/freq1),                  /* after half period */
     DS4002_LOW,                                  /* set output low */
     0,                              /* do not trigger or interrupt */
     DS4002_GOTO,               /* goto entry point (= first state) */
     0);                           /* no loop counter or jump value */
  ds4002_load_states(DS4002_1_BASE, 1);
                                  /* download program for channel 1 */
  /* ch2: ftod */
  ds4002_output_init();                /* prepare program variables */
  ds4002_define_entry();             /* entry point = program start */
  ds4002_define_state(
     DS4002_DELAY(0.5/freq2),                  /* after half period */
     DS4002_HIGH,                                /* set output high */
     DS4002_INTERRUPT,                            /* host interrupt */
     DS4002_CONTINUE,                   /* continue with next state */
     0);                           /* no loop counter or jump value */
  ds4002_define_state(
     DS4002_DELAY(0.5/freq2),                  /* after half period */
     DS4002_LOW,                                  /* set output low */
     DS4002_INTERRUPT,                            /* host interrupt */
     DS4002_GOTO,               /* goto entry point (= first state) */
     0);                           /* no loop counter or jump value */
  ds4002_load_states(DS4002_1_BASE, 2);
                                  /* download program for channel 2 */
  /* init channels 3 to 8 for input mode */
  ds4002_read_init(DS4002_1_BASE, 3, DS4002_RISING, 20);
  ds4002_read_init(DS4002_1_BASE, 4, DS4002_RISING, 20);
  ds4002_read_init(DS4002_1_BASE, 5, DS4002_RISING, 20);
  ds4002_read_init(DS4002_1_BASE, 6, DS4002_RISING, 20);
  ds4002_read_init(DS4002_1_BASE, 7, DS4002_RISING, 20);
  ds4002_read_init(DS4002_1_BASE, 8, DS4002_RISING, 20);
  RTLIB_SRT_START(0.001, isr_t1); /* initialize sampling clock timer */
  ds4002_enable_filter(DS4002_1_BASE);
  ds4002_start_channels(DS4002_1_BASE,     /* start channels 1 and 2 */
     DS4002_MASK(1) + DS4002_MASK(2) );
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

References

ds4002_define_entry	89
ds4002_define_state	85
DS4002_DELAY	97
ds4002_enable_filter	163
DS4002_EXEC_CMD	98
ds4002_init	19
DS4002_INT_CLEAR	16
ds4002_load_states	90
DS4002_MASK	101
ds4002_output_init	84
ds4002_read_contiguous	135
ds4002_read_init	129
ds4002_start_channels	91
ds4002_update_state	92

Macros

Where to go from here

Information in this section

Base Address of the I/O Board

DSxxxx_n_BASE Macros

When using I/O board functions, you always need the board's base address as a parameter. This address can easily be obtained by using the `DSxxxx_n_BASE` macros, where `DSxxxx` is the board name (for example, DS2001) and `n` is an index which counts boards of the same type. The board with the lowest base address is given index 1. The other boards of the same type are given consecutive numbers in order of their base addresses.

The macros reference an internal data structure which holds the addresses of all I/O boards in the system. The initialization function of the processor board (named `init`) creates this data structure. Hence, when you change an I/O board base address, it is not necessary to recompile the code of your application. For more information on the processor board's initialization function, refer to ds1006_init (DS1006 RTLib Reference 🕮) or init (DS1007 RTLib Reference 🕮).

Note

The `DSxxxx_n_BASE` macros can be used only after the processor board's initialization function `init` is called.

Example

This example demonstrates the use of the `DSxxxx_n_BASE` macros. There are two DS2001 boards, two DS2101 boards, and one DS2002 board connected to a PHS bus. Their base addresses have been set to different addresses. The following table shows the I/O boards, their base addresses, and the macros which can be used as base addresses:

Board	Base Address	Macro
DS2001	00H	`DS2001_1_BASE`
DS2002	20H	`DS2002_1_BASE`
DS2101	80H	`DS2101_1_BASE`
DS2001	90H	`DS2001_2_BASE`
DS2101	A0H	`DS2101_2_BASE`

DS4002_INT_CLEAR

Syntax

```
void DS4002_INT_CLEAR(phs_addr_t base, long mask)
```

Include file

`ds4002.h`

Purpose

To acknowledge interrupt requests from specified channels.

Description

For further information on using the interrupts, refer to Interrupts Provided by the DS4002 (DS4002 Features 🕮).

Note

Do not use this macro, if your application contains interrupts provided by the RTI library, i.e. a read event interrupt. For further information, refer to Interrupts (DS4002 RTI Reference 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

mask Specifies the channels from which the interrupt requests are acknowledged. You can use the mask defines `DS4002_MASK(1)` … `DS4002_MASK(8)`.

Return value

None

Example

This example shows how to acknowledge interrupt requests from channels 1 and 3.

```
DS4002_INT_CLEAR(DS4002_1_BASE, DS4002_MASK(1) + DS4002_MASK(3));
```

Related topics

References

DS4002_INT_STATUS

Syntax

```
long DS4002_INT_STATUS(phs_addr_t base)
```

Include file

ds4002.h

Purpose

To check which channel has generated an interrupt in input mode.

Description

For further information on using the interrupts, refer to Interrupts Provided by the DS4002 (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

This macro returns the interrupt status field in unsigned long format. If there is a channel interrupt request, the corresponding bit is set in the interrupt status field (channel 1 = bit 0, channel 2 = bit 1, ...).

Example

This example shows how to check channel 1 for interrupt generation.

```
if (DS4002_INT_STATUS(DS4002_1_BASE) && DS4002_MASK(1) )
...
```

Related topics

References

Base Address of the I/O Board.. 15
ds4002_d2f_int_init.. 55
ds4002_delayed_mono_int_init... 64
DS4002_INT_CLEAR.. 16
DS4002_MASK.. 101
ds4002_pwm_int_init.. 37

Initialization

Introduction

Before you can use the DS4002, you have to perform the initialization process.

ds4002_init

Syntax

```
int ds4002_init(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To initialize the DS4002 board.

Description

The `ds4002_init` function carries out the following initialization steps:

1. The function allocates dynamic memory for internal data storage.
2. Signal filtering is disabled.
3. All DS4002 registers are set to their initial values.
4. The board controller is reset.
5. All channels are set to input mode.
6. Capture and edge detection is disabled on all channels.
7. All digital I/O groups are set to non strobed input.

Note

- The initialization function of the processor board must be called before the `DS4002_init` function.
- The `DS4002_init` function must be called before any other DS4002 function can be used.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

Returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4002_ALLOC_ERROR`	The allocation of some dynamic memory failed. The application terminates.
`DS4002_NO_ERROR`	Initialization correctly executed.

Messages

The following messages are defined:

ID	Type	Message	Description
201	Error	ds4002_init(): Invalid PHS-bus base 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.
-174	Error	ds4002_init(0x??): Board not found!	No DS4002 board could be found at the specified PHS-bus address. Check if the DSxxxx_n_BASE macro corresponds to the I/O board used.
-187	Error	ds4002_init(0x??): Memory allocation error!	The allocation of some dynamic memory for internal data storage has failed.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to initialize a DS4002 at address `DS4002_1_BASE`:

```
void main(void)
{
   init();
   ds4002_init(DS4002_1_BASE);
   …
}
```

Related topics

References

Base Address of the I/O Board.. 15

Digital I/O Unit

Introduction

The DS4002 board provides 24 bidirectional plus 4 input and 4 output TTL digital I/O lines. With the RTLib functions you can program the digital I/O unit.

Where to go from here

Information in this section

ds4002_dio_init

Syntax

```
void ds4002_dio_init(
   phs_addr_t base,
   long iomode)
```

Include file

`ds4002.h`

Purpose

To initialize the DS4002 digital parallel I/O port.

Description

The DS4002 digital parallel I/O port is initialized as follows:

- D27 … 24 is always in output mode, D31 … 28 is always in normal input mode.
- If no constant is selected for a bit group, it is is configured as normal input.
- In strobed input mode the strobe input PSTB must be connected. Input data is latched with the rising edge of PSTB.
- When reading from the parallel I/O port, a 1 µs low pulse is generated at output PACK.
- When writing to the parallel I/O port, a 1 µs low pulse is generated at output PRDY.

For information on PSTB, PACK and PRDY, refer to Strobing Inputs (DS4002 Features 🕮).

> **Note**
>
> The `ds4002_init` function must be called before this function can be used.

I/O mapping

For information on the I/O mapping, refer to Digital I/O Unit (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

iomode Selects the I/O data direction and input mode. The following symbols are predefined:

Symbol	Related Bit Group
Normal input mode	
`DS4002_IN_0`	D7 … 0
`DS4002_IN_1`	D15 … 8
`DS4002_IN_2`	D23 … 16
Strobed input mode	
`DS4002_STRB_IN_0`	D7 … 0
`DS4002_STRB_IN_1`	D15 … 8
`DS4002_STRB_IN_2`	D23 … 16

Symbol	Related Bit Group
Output mode	
`DS4002_OUT_0`	D7 … 0
`DS4002_OUT_1`	D15 … 8
`DS4002_OUT_2`	D23 … 16

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_dio_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to set D7…0 to normal input mode, D15…8 to output mode and D23…16 to strobed input mode:

```
ds4002_dio_init(DS4002_1_BASE, DS4002_IN_0 | DS4002_OUT_1 |
   DS4002_STRB_IN_2);
```

Related topics

References

Base Address of the I/O Board 15
ds4002_dio_initialize 23
ds4002_init 19
Macros 15

ds4002_dio_initialize

Syntax

```
__INLINE void ds4002_dio_initialize(
   phs_addr_t base,
   long iomode,
   long data);
```

Include file

`ds4002.h`

Purpose

To initialize the DS4002 digital parallel I/O port with predefined output values.

Description

The DS4002 digital parallel I/O port is initialized as follows:

- D27 … 24 is always in output mode, D31 … 28 is always in normal input mode.
- If no constant is selected for a bit group, then this group will be configured as normal input.
- In strobed input mode the strobe input PSTB must be connected. Input data is latched with the rising edge of PSTB.
- When reading from the parallel I/O port, a 1 µs low pulse is generated at output PACK.
- When writing to the parallel I/O port, a 1 µs low pulse is generated at output PRDY.

For information on PSTB, PACK and PRDY, refer to Strobing Inputs (DS4002 Features 📖).

While the `ds4002_dio_init` function sets the output pins to low level by default, you can use the `ds4002_dio_initialize` function also to initialize the output values with high level. This can be necessary for sensitive hardware to avoid low level peaks, which will appear when the pin is set to low level value during initialization and afterwards changing it to high level value by using the `ds4002_dio_bit_out` function.

> **Note**
>
> The `ds4002_init` function must be called before this function can be used.

I/O mapping

For information on the I/O mapping, refer to Digital I/O Unit (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

iomode Selects the I/O data direction and input mode. The following symbols are predefined:

Symbol	Related Bit Group
Normal input mode	
`DS4002_IN_0`	D7 … 0
`DS4002_IN_1`	D15 … 8
`DS4002_IN_2`	D23 … 16
Strobed input mode	
`DS4002_STRB_IN_0`	D7 … 0
`DS4002_STRB_IN_1`	D15 … 8
`DS4002_STRB_IN_2`	D23 … 16

Symbol	Related Bit Group
Output mode	
`DS4002_OUT_0`	D7 … 0
`DS4002_OUT_1`	D15 … 8
`DS4002_OUT_2`	D23 … 16

data Specifies initial values for pins configured as output (`0x00000000` … `0x0FFFFFFF`).

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_dio_initialize(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to set D7 … 0 to normal input mode, D15 … 8 to output mode and D23 … 16 to strobed input mode:

```
ds4002_dio_initialize(DS4002_1_BASE, DS4002_IN_0 |
   DS4002_OUT_1 | DS4002_STRB_IN_2, data);
```

The I/O pins specified as outputs are initialized with the data specified in the `data` variable, before setting the pin as output.

Related topics

References

ds4002_dio_bit_in

Syntax

```
UInt32 ds4002_dio_bit_in(
   phs_addr_t base,
   long mask)
```

Include file

`ds4002.h`

Purpose

To read the parallel I/O port.

Description

This function reads the state of the digital I/O pins specified as inputs. For pins which are in output mode, the last data which was written is returned.

> **Note**
>
> - The port must have been initialized by using `ds4002_dio_init` or `ds4002_dio_initialize`.
> - D27 … 24 is always in output mode, D31 … 28 is always in normal input mode.

I/O mapping

For information on the I/O mapping, refer to Digital I/O Unit (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

mask Specifies a mask for the bits to be read.

Return value

Returns the value of the I/O pins that have been specified by the `mask` parameter.

Execution times

For information, refer to Function Execution Times on page 167.

Example

The following example shows how to use the function:

```
UInt32 data;
/* set I/O group 0 to output, groups 1 and 2 are inputs    */
ds4002_dio_init(DS4002_1_BASE, DS4002_OUT_0);
/* read input data only of I/O pins 4 and 7                */
data = ds4002_dio_bit_in(DS4002_1_BASE, 0x00000090);
```

Related topics

References

Base Address of the I/O Board	15
ds4002_dio_bit_out	27
ds4002_dio_init	21
ds4002_dio_initialize	23
ds4002_in32	28
Macros	15

ds4002_dio_bit_out

Syntax

```
void ds4002_dio_bit_out(
   phs_addr_t base,
   long mask,
   UInt32 data)
```

Include file

`ds4002.h`

Purpose

To write data to the parallel I/O port.

Description

This function writes the `data` parameter to the output bits. Data for bits which are configured as input bits is ignored.

Only data bits specified by the corresponding bits in the `mask` parameter are affected. All other output bits stay at their previous state.

Note

- The port must have been initialized by using `ds4002_dio_init` or `ds4002_dio_initialize`.
- D27 … 24 is always in output mode, D31 … 28 is always in normal input mode.

I/O mapping

For information on the I/O mapping, refer to Digital I/O Unit (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

mask Masks the bits to be written.

data Specifies the data to be written.

Execution times

For information, refer to Function Execution Times on page 167.

Example

The following example shows how to use the function:

```
UInt32 data;
/* set I/O group 0 to output, groups 1 and 2 are inputs     */
ds4002_dio_init(DS4002_1_BASE, DS4002_OUT_0);
```

```
/* write output data only to pins 1 and 2,                  */
/* other pins remain unchanged                              */
/* pin 1 is set and pin 2 is cleared                        */
ds4002_dio_bit_out(DS4002_1_BASE, 0x00000006, 0x00000002);
```

Related topics

References

Base Address of the I/O Board.......... 15
ds4002_dio_bit_in.......... 25
ds4002_dio_init.......... 21
ds4002_dio_initialize.......... 23
ds4002_out32.......... 29
Macros.......... 15

ds4002_in32

Syntax

```
UInt32 ds4002_in32(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To read data from the parallel I/O port.

Description

This function reads the input bits. For bits which are in output mode, the last data which was written is returned.

> **Note**
>
> - The port must have been initialized by using **`ds4002_dio_init`** or **`ds4002_dio_initialize`**.
> - D27 … 24 is always in output mode, D31 … 28 is always in normal input mode.

I/O mapping

For information on the I/O mapping, refer to Digital I/O Unit (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

Returns the data of the parallel I/O port.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

References

Base Address of the I/O Board	15
ds4002_dio_bit_in	25
ds4002_dio_init	21
ds4002_dio_initialize	23
ds4002_out32	29
Macros	15

ds4002_out32

Syntax

```
void ds4002_out32(
   phs_addr_t base,
   UInt32 data)
```

Include file

`ds4002.h`

Purpose

To write data to the parallel I/O port.

Description

This function writes the `data` parameter to the output bits. Data for bits which are configured as input bits is ignored.

> **Note**
>
> - The port must have been initialized by using `ds4002_dio_init` or `ds4002_dio_initialize`.
> - D27…24 is always in output mode, D31…28 is always in normal input mode.

I/O mapping

For information on the I/O mapping, refer to Digital I/O Unit (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

data Specifies the data to be written.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

References

Base Address of the I/O Board 15
ds4002_dio_init 21
ds4002_dio_initialize 23
ds4002_in32 28
Macros 15

Timing I/O Unit

Where to go from here

Information in this section

1-Phase PWM Signal Generation (PWM)

Introduction

The timing I/O unit of the DS4002 provides outputs for 1-phase PWM signal generation on up to 8 channels. The polarity of the 1-phase PWM signals is active high.

Where to go from here

Information in this section

Example of Using the 1-Phase Signal Generation Functions

Introduction

The following example demonstrates how to use the PWM functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 4 are initialized for PWM generation. Channels 5 … 8 are initialized for duty cycle and frequency measurement. All channels are updated or read in an interrupt service routine every 100 µs.

You have to connect the channels as follows:

Connect …	With …
Channel 1	Channel 5
Channel 2	Channel 4

Connect ...	With ...
Channel 3	Channel 7
Channel 4	Channel 8

```
#include "brtenv.h"
#include "ds4002.h"
/************************************************************************
  global variables
************************************************************************/
dsfloat freq1 = 60000.0;
dsfloat freq2 = 61000.0;
dsfloat freq3 = 62000.0;
dsfloat freq4 = 63000.0;
dsfloat duty1 = 0.1;
dsfloat duty2 = 0.4;
dsfloat duty3 = 0.6;
dsfloat duty4 = 0.9;
dsfloat freq5 = 0.0;
dsfloat freq6 = 0.0;
dsfloat freq7 = 0.0;
dsfloat freq8 = 0.0;
dsfloat duty5 = 0.0;
dsfloat duty6 = 0.0;
dsfloat duty7 = 0.0;
dsfloat duty8 = 0.0;
/********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long count, len;
  ts_timestamp_type ts;
  ds4002_pwm_update(DS4002_1_BASE, 1, 1/freq1, duty1);
  ds4002_pwm_update(DS4002_1_BASE, 2, 1/freq2, duty2);
  ds4002_pwm_update(DS4002_1_BASE, 3, 1/freq3, duty3);
  ds4002_pwm_update(DS4002_1_BASE, 4, 1/freq4, duty4);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 5, count, &len, &freq5, &duty5);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 6, count, &len, &freq6, &duty6);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 7, count, &len, &freq7, &duty7);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 8, count, &len, &freq8, &duty8);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                        /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);         /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_pwm_init( DS4002_1_BASE, 1, 1/freq1, duty1);
  ds4002_pwm_init( DS4002_1_BASE, 2, 1/freq2, duty2);
  ds4002_pwm_init( DS4002_1_BASE, 3, 1/freq3, duty3);
  ds4002_pwm_init( DS4002_1_BASE, 4, 1/freq4, duty4);
  ds4002_pwm2d_init(DS4002_1_BASE, 5, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 6, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 7, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 8, 0, 0.0);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_pwm_init

Syntax

```
void ds4002_pwm_init(
   phs_addr_t base,
   long channel,
   dsfloat tp,
   dsfloat duty)
```

Include file

`ds4002.h`

Purpose

To initialize a channel for PWM generation.

Description

After initialization, channel operation is started. PWM parameters may be updated by using the `ds4002_pwm_update` function.

I/O mapping

For information on the I/O mapping, refer to 1-Phase PWM Signal Generation (PWM) (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

tp Specifies the period of one PWM cycle. Depending on the number of active DS4002 channels, a minimum period of 1.2 µs … 8 µs must be given. The period may be as long as 107 s.

duty Specifies the duty cycle within the range 0.0 … 1.0.

Note

- Due to the limitations of the DS4002 the minimum width of the low or high part of the PWM signal is 600 ns. High part pulse widths below 600 ns will result in duty cycle = 0 (permanently low), low part pulse widths below 600 ns will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used. For further information, refer to 1-Phase PWM Signal Generation (PWM) (DS4002 Features 🕮).

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_pwm_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the 1-Phase Signal Generation Functions.......... 33

References

Base Address of the I/O Board.......... 15
ds4002_init.......... 19
ds4002_pwm_update.......... 39
Macros.......... 15

ds4002_pwm_int_init

Syntax

```
void ds4002_pwm_int_init(
   phs_addr_t base,
   long channel,
   dsfloat tp,
   dsfloat duty,
   long intgen)
```

Include file

`ds4002.h`

Purpose

To initialize the specified channel for PWM generation with interrupt generation.

Description

After initialization, channel operation is started. If interrupt generation is enabled by the `intgen` parameter, on each rising or falling edge an interrupt is generated.

I/O mapping

For information on the I/O mapping, refer to 1-Phase PWM Signal Generation (PWM) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number 1 … 2.

> **Note**
>
> Only channel 1 and 2 are supporting interrupt generation! If interrupt generation is enabled and channel 3 … 8 are specified, the function exits with an error message.

tp Specifies the PWM-signal period in seconds within the range 1.2e-6 … 107.0. Depending on the number of active DS4002 channels, a minimum period of 1.2 µs to 8 µs must be given. The period may be as long as 107 s.

duty Specifies the duty cycle within the range 0.0 … 1.0.

Note

- Due to the limitations of the DS4002 the minimum width of the low or high part of the PWM signal is 600 ns. High part pulse widths below 600 ns will result in duty cycle = 0 (permanently low), low part pulse widths below 600 ns will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used. For further information, refer to 1-Phase PWM Signal Generation (PWM) (DS4002 Features 📖).

intgen Enables the interrupt generation. The following symbols are predefined:

Symbol	Meaning
`DS4002_INT_NONE`	No interrupts
`DS4002_INT_RISING`	Interrupt on rising edge
`DS4002_INT_FALLING`	Interrupt on falling edge

Note

You must not combine the symbols `DS4002_INT_RISING` and `DS4002_INT_FALLING`.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds4002_pwm_int_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.
-198	Error	ds4002_pwm_int_init(0x??): Can't generate interrupts on channel ?!	The specified channel is unable to generate interrupts. Only channel 1 and 2 can generate interrupts in output mode.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

1-Phase PWM Signal Generation (PWM) (DS4002 Features 📖)

References

Base Address of the I/O Board 15
ds4002_init 19
DS4002_INT_CLEAR 16
DS4002_INT_STATUS 17
ds4002_pwm_update 39

ds4002_pwm_update

Syntax

```
void ds4002_pwm_update(
   phs_addr_t base,
   long channel,
   dsfloat tp,
   dsfloat duty)
```

Include file

`ds4002.h`

Purpose

To update PWM parameters.

Description

The PWM parameters of the specified channel are updated. The period of one PWM cycle is given by the `tp` parameter. Depending on the number of active DS4002 channels, a minimum period of 1.2 µs … 8 µs must be given. The period may be as long as 107 s. The duty cycle is given by the `duty` parameter. Updates will become effective with the next PWM cycle, starting with the low period (block update mode). For further information, refer to Updating State Parameters (DS4002 Features 📖).

Note

- The specified channel must have been initialized by using `ds4002_pwm_init` or `ds4002_pwm_int_init`.
- Due to the limitations of the DS4002 the minimum width of the low or high part of the PWM signal is 600 ns. High part pulse widths below 600 ns will result in duty cycle = 0 (permanently low), low part pulse widths below 600 ns will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used. For further information, refer to 1-Phase PWM Signal Generation (PWM) (DS4002 Features 🕮).

I/O mapping

For information on the I/O mapping, refer to 1-Phase PWM Signal Generation (PWM) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

tp Specifies the period of one PWM cycle within the range 1.2e-6 … 107.0.

duty Specifies the duty cycle within the range 0.0 … 1.0.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the 1-Phase Signal Generation Functions.. 33

References

Base Address of the I/O Board.. 15
ds4002_pwm_init... 35
ds4002_pwm_int_init... 37
Macros... 15

3-Phase PWM Signal Generation (PWM3)

Introduction

The timing I/O unit of the DS4002 provides outputs for 3-phase PWM signal generation. PWM3 signals are centered around the middle of the PWM period. The polarity of the PWM3 signals is active high.

Where to go from here

Information in this section

Example of Using the 3-Phase Signal Generation Functions

Introduction

The following example demonstrates how to use the PWM3 functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 3 are initialized for 3-phase PWM generation. Channels 5 … 7 are initialized for duty cycle and frequency measurement. All channels are updated or read in an interrupt service routine every 100 µs.

You have to connect the channels as follows:

Connect ...	With ...
Channel 1	Channel 5
Channel 2	Channel 6
Channel 3	Channel 7

```
#include "brtenv.h"
#include "ds4002.h"
/***********************************************************************
  global variables
************************************************************************/
dsfloat freq  = 60000.0;
dsfloat duty1 = 0.1;
dsfloat duty2 = 0.4;
dsfloat duty3 = 0.6;
dsfloat freq5 = 0.0;
dsfloat freq6 = 0.0;
dsfloat freq7 = 0.0;
dsfloat duty5 = 0.0;
dsfloat duty6 = 0.0;
dsfloat duty7 = 0.0;
/********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long count;
  ts_timestamp_type ts;
  ds4002_pwm3_update(DS4002_1_BASE, 1, 2, 3, 1/freq, duty1, duty2, duty3);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 5, count, &len, &freq5, &duty5);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 6, count, &len, &freq6, &duty6);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 7, count, &len, &freq7, &duty7);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                          /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);           /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_pwm3_init( DS4002_1_BASE, 1, 2, 3, 1/freq, duty1, duty2, duty3);
  ds4002_pwm2d_init(DS4002_1_BASE, 5, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 6, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 7, 0, 0.0);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_pwm3_init

Syntax

```
void ds4002_pwm3_init(
   phs_addr_t base,
   long ch1,
   long ch2,
   long ch3,
   dsfloat tp,
   dsfloat duty1,
   dsfloat duty2,
   dsfloat duty3)
```

Include file

`ds4002.h`

Purpose

To initialize the specified channels for 3-phase PWM generation.

Description

After initialization, channel operation is started. Any combination of channels is valid, as long as 3 different channels are selected. The period of one PWM cycle is given by the parameter `tp`. Depending on the number of active DS4002 channels, a minimum period of 4 µs … 8 µs must be given. The period may be as long as 107 s. The duty cycles for the 3 output channels are given by the parameters **`duty1`**, **`duty2`** and **`duty3`** and may range from 0 … 1.

Note

- Use the **`ds4002_pwm3_update`** function to update the PWM parameters.
- Due to the limitations of the DS4002 the minimum width of the low or high part of the 3-phase PWM signal is 1.4 µs. High part pulse widths below 1.4 µs will result in duty cycle = 0 (permanently low), low part pulse widths below 1.4 µs will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used.

For further information, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮).

I/O mapping

For information on the I/O mapping, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

ch1 Specifies the logical number of channel 1 within the range 1 … 8. It must be different from `ch2` and `ch3`.

ch2 Specifies the logical number of channel 2 within the range 1 … 8. It must be different from `ch1` and `ch3`.

ch3 Specifies the logical number of channel 3 within the range 1 … 8. It must be different from `ch1` and `ch2`.

tp Specifies the period of one PWM cycle in seconds within the range 4 µs … 107 s. Depending on the number of active DS4002 channels, a minimum period of 4 … 8 µs must be given. The period may be as long as 107 s.

duty1 Specifies the duty cycle of channel 1 in the range 0.0 … 1.0.

duty2 specifies the duty cycle of channel 2 in the range 0.0 … 1.0.

duty3 Specifies the duty cycle of channel 3 in the range 0.0 … 1.0.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_pwm3_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

3-Phase PWM Signal Generation (PWM3) (DS4002 Features 📖)

Examples

Example of Using the 3-Phase Signal Generation Functions..41

References

Base Address of the I/O Board..15
ds4002_init...19
ds4002_pwm3_update...45
Macros...15

ds4002_pwm3_update

Syntax

```
void ds4002_pwm3_update(
   phs_addr_t base,
   long ch1,
   long ch2,
   long ch3,
   dsfloat tp,
   dsfloat duty1,
   dsfloat duty2,
   dsfloat duty3)
```

Include file

`ds4002.h`

Purpose

To update the PWM parameters of 3 channels.

Description

Updates will become effective synchronously for all 3 phases with the next PWM cycle (Synchronous update mode, refer to Updating State Parameters (DS4002 Features 🕮)).

The period of one PWM cycle is given by the `tp` parameter. Depending on the number of active DS4002 channels, a minimum period of 4 … 8 µs must be given. The period may be as long as 107 s.

The duty cycles for the 3 output channels are given by the `duty1`, `duty2` and `duty3` parameters.

Note

- The channels must have been initialized by using `ds4002_pwm3_init`.
- Due to the limitations of the DS4002 the minimum width of the low or high part of the 3-phase PWM signal is 1.4 µs. High part pulse widths below 1.4 µs will result in duty cycle = 0 (permanently low), low part pulse widths below 1.4 µs will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used. For further information, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮).

I/O mapping

For information on the I/O mapping, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

ch1 Specifies the logical number of channel 1 within the range 1 … 8.

ch2 Specifies the logical number of channel 2 within the range 1 … 8.

ch3 Specifies the logical number of channel 3 within the range 1 … 8.

tp Specifies the PWM signal period in seconds within the range 4e-6 … 107.0.

duty1 Specifies the duty cycle of channel 1 in the range 0.0 … 1.0.

duty2 specifies the duty cycle of channel 2 in the range 0.0 … 1.0.

duty3 Specifies the duty cycle of channel 3 in the range 0.0 … 1.0.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

3-Phase PWM Signal Generation (PWM3) (DS4002 Features 📖)
Updating State Parameters (DS4002 Features 📖)

Examples

Example of Using the 3-Phase Signal Generation Functions	41

References

Base Address of the I/O Board	15
ds4002_pwm3_init	43
Macros	15

ds4002_pwm3_int_init

Syntax

```
void ds4002_pwm3_int_init (
   phs_addr_t base,
   long ch1,
   long ch2,
   long ch3,
   dsfloat tp,
   dsfloat duty1,
   dsfloat duty2,
   dsfloat duty3,
   long intgen)
```

Include file

`ds4002.h`

Purpose

To initialize channels for 3-phase PWM generation.

Description

After initialization, channel operation is started. Any combination of channels is valid, as long as 3 different channels are selected. The period of one PWM cycle is given by the parameter `tp`. Depending on the number of active DS4002 channels, a minimum period of 5 … 8 µs must be given. The period may be as long as 107 s. The duty cycles for the 3 output channels are given by the parameters `duty1`, `duty2` and `duty3` and may range from 0 … 1.

If interrupt generation is enabled by the `intgen` parameter, on each middle of the high or the low period of the `ch1` PWM signal an interrupt is generated.

When generating PWM signals with high frequency, the IRQ during the high period may not occur exactly in the middle. The deviation is 200 ns.

Note

- If interrupt generation is enabled, only channel 1 and 2 are valid values for the parameter `ch1`. If one of the channels 3 … 8 is specified for `ch1`, the function exits with an error message.
- Use the `ds4002_pwm3_int_update` function to update the PWM parameters.
- Due to the limitations of the DS4002 the minimum width of the low or high part of the 3-phase PWM signal is 1.4 µs. High part pulse widths below 1.4 µs will result in duty cycle = 0 (permanently low), low part pulse widths below 1.4 µs will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used.

For further information, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮).

I/O mapping

For information on the I/O mapping, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

ch1 Specifies the logical number of channel 1. It must be different from `ch2` and `ch3`.

ch2 Specifies the logical number of channel 2 within the range 1 … 8. It must be different from `ch1` and `ch3`.

ch3 Specifies the logical number of channel 3 within the range 1 … 8. It must be different from `ch1` and `ch2`.

tp Specifies the PWM signal period in seconds within the range 5e-6 … 107.0.

duty1 Specifies the duty cycle of channel 1 in the range 0.0 … 1.0.

duty2 specifies the duty cycle of channel 2 in the range 0.0 … 1.0.

duty3 Specifies the duty cycle of channel 3 in the range 0.0 … 1.0.

intgen Enables interrupt generation. The following symbols are predefined:

Symbol	Meaning
`DS4002_INT_NONE`	No interrupts
`DS4002_INT_HIGH`	Interrupt on the middle of the high period
`DS4002_INT_LOW`	Interrupt on the middle of the low period

Note

You cannot combine the symbols `DS4002_INT_HIGH` and `DS4002_INT_LOW`.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds4002_pwm3_int_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.
-198	Error	ds4002_pwm3_int_init(0x??): Can’t generate interrupts on channel ?!	The specified 'ch1' channel is unable to generate interrupts. Only channel 1 and 2 can generate interrupts in output mode.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮)

References

Base Address of the I/O Board........ 15
ds4002_init........ 19
DS4002_INT_CLEAR........ 16
DS4002_INT_STATUS........ 17
ds4002_pwm3_int_update........ 49
ds4002_pwm3_update........ 45

ds4002_pwm3_int_update

Syntax

```
void ds4002_pwm3_int_update (
   phs_addr_t base,
   long ch1,
   long ch2,
   long ch3,
   dsfloat tp,
   dsfloat duty1,
   dsfloat duty2,
   dsfloat duty3)
```

Include file

`ds4002.h`

Purpose

To update the PWM parameters for the 3-phase signal generation with interrupt generation.

Description

Updates will become effective synchronously for all 3 phases with the next PWM cycle (Synchronous update mode, refer to Updating State Parameters (DS4002 Features 🕮)).

The period of one PWM cycle is given by the `tp` parameter. Depending on the number of active DS4002 channels, a minimum period of 5 … 8 µs must be given. The period may be as long as 107 s.

The duty cycles for the 3 output channels are given by the `duty1`, `duty2` and `duty3` parameters.

Note

- The channels must have been initialized by using `ds4002_pwm3_int_init`.
- Due to the limitations of the DS4002 the minimum width of the low or high part of the 3-phase PWM signal is 1.4 µs. High part pulse widths below 1.4 µs will result in duty cycle = 0 (permanently low), low part pulse widths below 1.4 µs will result in duty cycle = 1 (permanently high).
- Depending on the number of active channels the PWM signal may become asynchronous or erroneous, if PWM periods below 8 µs are used. For further information, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 📖).

I/O mapping

For information on the I/O mapping, refer to 3-Phase PWM Signal Generation (PWM3) (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

ch1 Specifies the logical number of channel 1 within the range 1 … 8.

ch2 Specifies the logical number of channel 2 within the range 1 … 8.

ch3 Specifies the logical number of channel 3 within the range 1 … 8.

tp Specifies the PWM signal period in seconds within the range 5e-6 … 107.0.

duty1 Specifies the duty cycle of channel 1 in the range 0.0 … 1.0.

duty2 specifies the duty cycle of channel 2 in the range 0.0 … 1.0.

duty3 Specifies the duty cycle of channel 3 in the range 0.0 … 1.0.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

3-Phase PWM Signal Generation (PWM3) (DS4002 Features 🕮)
Updating State Parameters (DS4002 Features 🕮)

References

Base Address of the I/O Board.. 15
ds4002_pwm3_int_init.. 46
Macros.. 15

Square-Wave Signal Generation (D2F)

Introduction

The timing I/O unit of the DS4002 provides outputs for square-wave signal generation (D2F) on up to 8 channels.

Where to go from here

Information in this section

Example of Using the Square-Wave Signal Generation Functions

Introduction

The following example demonstrates how to use the D2F functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 4 are initialized for generating square-wave signals. Channels 5 … 8 are initialized for measuring square-wave signals. All channels are updated or read in an interrupt service routine every 1 ms.

You have to connect the channels as follows:

Connect …	With …
Channel 1	Channel 5
Channel 2	Channel 6

Connect ...	With ...
Channel 3	Channel 7
Channel 4	Channel 8

```
#include "brtenv.h"
#include "ds4002.h"
/************************************************************************
  global variables
************************************************************************/
dsfloat freq1 = 41000.0;
dsfloat freq2 = 42000.0;
dsfloat freq3 = 43000.0;
dsfloat freq4 = 44000.0;
dsfloat freq5 = 0.0;
dsfloat freq6 = 0.0;
dsfloat freq7 = 0.0;
dsfloat freq8 = 0.0;
long ch5_error = 0;
long ch6_error = 0;
long ch7_error = 0;
long ch8_error = 0;
  ds4002_d2f_update(DS4002_1_BASE, 1, freq1);
  ds4002_d2f_update(DS4002_1_BASE, 2, freq2);
  ds4002_d2f_update(DS4002_1_BASE, 3, freq3);
  ds4002_d2f_update(DS4002_1_BASE, 4, freq4);
  /* with this service routine called every 1ms, this should process
 all incoming data up to 100kHz */
  count = 100;
  ch5_error = ds4002_f2d_contig(DS4002_1_BASE, 5, count, &len, &freq5);
  count = 100;
  ch6_error = ds4002_f2d_contig(DS4002_1_BASE, 6, count, &len, &freq6);
  count = 1;
  ch7_error = ds4002_f2d_overl(DS4002_1_BASE, 7, count, &len, &freq7);
  count = 1;
  ch8_error = ds4002_f2d_overl(DS4002_1_BASE, 8, count, &len, &freq8);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                          /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);           /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_d2f_init(DS4002_1_BASE, 1, freq1);
  ds4002_d2f_init(DS4002_1_BASE, 2, freq2);
  ds4002_d2f_init(DS4002_1_BASE, 3, freq3);
  ds4002_d2f_init(DS4002_1_BASE, 4, freq4);
  ds4002_f2d_init(DS4002_1_BASE, 5, 0, 0.0);
  ds4002_f2d_init(DS4002_1_BASE, 6, 0, 0.0);
  ds4002_f2d_init(DS4002_1_BASE, 7, 0, 0.0);
  ds4002_f2d_init(DS4002_1_BASE, 8, 0, 0.0);
  RTLIB_SRT_START(0.001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_d2f_init

Syntax

```
void ds4002_d2f_init(
   phs_addr_t base,
   long channel,
   dsfloat freq)
```

Include file

`ds4002.h`

Purpose

To initialize the specified channel for square-wave frequency generation.

Description

After initialization, channel operation is started. The frequency is given by the `freq` parameter. Depending on the number of active DS4002 channels, you can specify a maximum frequency of 125 kHz … 833 kHz. The frequency may be as low as 0.01 Hz. The frequency may be updated by using the `ds4002_d2f_update` function.

I/O mapping

For information on the I/O mapping, refer to Generation of Simple Signals (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

freq Specifies the frequency within the range 0.01 … 833.0 kHz.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_d2f_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the Square-Wave Signal Generation Functions.. 52

References

Base Address of the I/O Board.. 15
ds4002_d2f_update.. 57
ds4002_init.. 19
Macros.. 15

ds4002_d2f_int_init

Syntax

```
void ds4002_d2f_int_init (
   phs_addr_t base,
   long channel,
   dsfloat freq,
   long intgen)
```

Include file

`ds4002.h`

Purpose

To initialize the specified channel for square-wave frequency generation with interrupt support.

Description

After initialization, channel operation is started. If interrupt generation is enabled, on the rising or falling edge of the signal an interrupt is generated. The frequency is given by the **`freq`** parameter. Depending on the number of active DS4002 channels, you can specify a maximum frequency of 125 kHz … 833 kHz must be given. The frequency may be as low as 0.01 Hz. The frequency may be updated by using the **`ds4002_d2f_update`** function.

I/O mapping

For information on the I/O mapping, refer to Generation of Simple Signals (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number within the range 1… 8.

> **Note**
>
> Only channel 1 and 2 are supporting interrupt generation. If interrupt generation is enabled and channel 3 … 8 are specified, the function exits with an error message.

freq Specifies the frequency within the range 0.01 … 833.0 kHz.

intgen Enables the interrupt generation. The following symbols are predefined:

Symbol	Meaning
`DS4002_INT_NONE`	No interrupts
`DS4002_INT_RISING`	Interrupt on rising edge
`DS4002_INT_FALLING`	Interrupt on falling edge

> **Note**
>
> You cannot combine several different symbols.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds4002_d2f_int_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the **`ds4002_init`** function.
-198	Error	ds4002_d2f_int_init(0x??): Can't generate interrupts on channel ?!	The specified channel is unable to generate interrupts. Only channel 1 and 2 can generate interrupts in output mode.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

References

Base Address of the I/O Board........ 15
ds4002_d2f_update........ 57
ds4002_init........ 19
DS4002_INT_CLEAR........ 16
DS4002_INT_STATUS........ 17

ds4002_d2f_update

Syntax

```
void ds4002_d2f_update(
   phs_addr_t base,
   long channel,
   dsfloat freq)
```

Include file

`ds4002.h`

Purpose

To update the frequency of a channel.

Description

Updates will become effective with the next cycle, starting with the low period (Block update mode, refer to Updating State Parameters (DS4002 Features 🕮)). The frequency is given by the `freq` parameter. Depending on the number of active DS4002 channels, you can specify a maximum frequency of 125 kHz … 833 kHz. The frequency may be as low as 0.01 Hz.

> **Note**
>
> The channel must have been initialized by using `ds4002_d2f_init` or `ds4002_d2f_int_init`.

I/O mapping

For information on the I/O mapping, refer to Generation of Simple Signals (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

freq Specifies the frequency within the range 0.01 … 833.0 kHz.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

- Example of Using the Square-Wave Signal Generation Functions.. 52

References

- Base Address of the I/O Board... 15
- ds4002_d2f_init... 54
- ds4002_d2f_int_init... 55
- Macros... 15

Monoflop Signal Generation

Introduction

The timing I/O unit of the DS4002 provides outputs for monoflop signal generation on up to 8 channels. After monoflop signal generation is triggered, a high-active single pulse is output at the specified channel.

Where to go from here

Information in this section

Example of Using the Monoflop Signal Generation Functions

Introduction

The following example demonstrates how to use the monoflop signal generation functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 4 are initialized for single pulse generation. Channels 5 … 8 are initialized for duty cycle and frequency measurement. All channels are updated or read in an interrupt service routine every 100 µs. Frequency measurement should yield the interrupt frequency, and duty cycle should yield the pulse width divided by 100 µs.

You have to connect the channels as follows:

Connect ...	With ...
Channel 1	Channel 5
Channel 2	Channel 6
Channel 3	Channel 7
Channel 4	Channel 8

```
#include "brtenv.h"
#include "ds4002.h"
/************************************************************************
  global variables
************************************************************************/
dsfloat pulse1 =  3e-6;
dsfloat pulse2 =  5e-6;
dsfloat pulse3 = 10e-6;
dsfloat pulse4 = 20e-6;
dsfloat freq5 = 0.0;
dsfloat freq6 = 0.0;
dsfloat freq7 = 0.0;
dsfloat freq8 = 0.0;
dsfloat duty5 = 0.0;
dsfloat duty6 = 0.0;
dsfloat duty7 = 0.0;
dsfloat duty8 = 0.0;
/*********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long count;
  ts_timestamp_type ts;
  ds4002_mono_update(DS4002_1_BASE, 1, pulse1);
  ds4002_mono_update(DS4002_1_BASE, 2, pulse2);
  ds4002_mono_update(DS4002_1_BASE, 3, pulse3);
  ds4002_mono_update(DS4002_1_BASE, 4, pulse4);
  ds4002_mono_start(DS4002_1_BASE, 1);
  ds4002_mono_start(DS4002_1_BASE, 2);
  ds4002_mono_start(DS4002_1_BASE, 3);
  ds4002_mono_start(DS4002_1_BASE, 4);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 5, count, &len, &freq5, &duty5);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 6, count, &len, &freq6, &duty6);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 7, count, &len, &freq7, &duty7);
  count = 1;
  ds4002_pwm2d_overl(DS4002_1_BASE, 8, count, &len, &freq8, &duty8);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
```

```
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                           /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);           /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_mono_init(DS4002_1_BASE, 1, pulse1);
  ds4002_mono_init(DS4002_1_BASE, 2, pulse2);
  ds4002_mono_init(DS4002_1_BASE, 3, pulse3);
  ds4002_mono_init(DS4002_1_BASE, 4, pulse4);
  ds4002_pwm2d_init(DS4002_1_BASE, 5, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 6, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 7, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 8, 0, 0.0);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_mono_init

Syntax

```
void ds4002_mono_init(
   phs_addr_t base,
   long channel,
   dsfloat tm)
```

Include file

`ds4002.h`

Purpose

To initialize the channel for pulse generation.

Description

The specified channel is initialized for pulse generation. After initialization, channel operation must be triggered by using `ds4002_mono_start`, an internal or an external trigger. For further information, refer to Monoflop Signal Generation (DS4002 Features 📖).

The pulse width is given by the `tm` parameter.

> **Note**
>
> - The pulse width may be updated by using the `ds4002_mono_update` function.
> - Depending on the number of active DS4002 channels, a minimum period of 0.6 … 4 µs must be selected. The period may be as long as 107 s.
>   For further information, refer to Monoflop Signal Generation (DS4002 Features 📖).

I/O mapping

For information on the I/O mapping, refer to Monoflop Signal Generation (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

tm Specifies the pulse width in seconds within the range 0.6 µs … 107.0 s.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_mono_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the Monoflop Signal Generation Functions.. 59

References

Base Address of the I/O Board.. 15
ds4002_init.. 19
ds4002_mono_start.. 68
ds4002_mono_update.. 63
Macros.. 15

ds4002_mono_update

Syntax

```
void ds4002_mono_update(
   phs_addr_t base,
   long channel,
   dsfloat tm)
```

Include file

`ds4002.h`

Purpose

To update the pulse width of a channel.

Description

Updates will be effective for the next pulse. The pulse width is given by the `tm` parameter.

> **Note**
>
> - The channel must have been initialized by using `ds4002_mono_init`.
> - Depending on the number of active DS4002 channels, a minimum period of 0.6 µs to 4 µs must be selected. The period may be as long as 107 s.
>   For further information, refer to Monoflop Signal Generation (DS4002 Features 🕮).

I/O mapping

For information on the I/O mapping, refer to Monoflop Signal Generation (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

tm Specifies the pulse width in seconds within the range 0.6 µs … 107.0 s.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

Monoflop Signal Generation (DS4002 Features 📖)

Examples

Example of Using the Monoflop Signal Generation Functions... 59

References

Base Address of the I/O Board... 15
ds4002_mono_init.. 61
Macros.. 15

ds4002_delayed_mono_int_init

Syntax

```
void ds4002_delayed_mono_int_init (
   phs_addr_t base,
   long channel,
   dsfloat td,
   dsfloat tm,
   long intgen)
```

Include file

`ds4002.h`

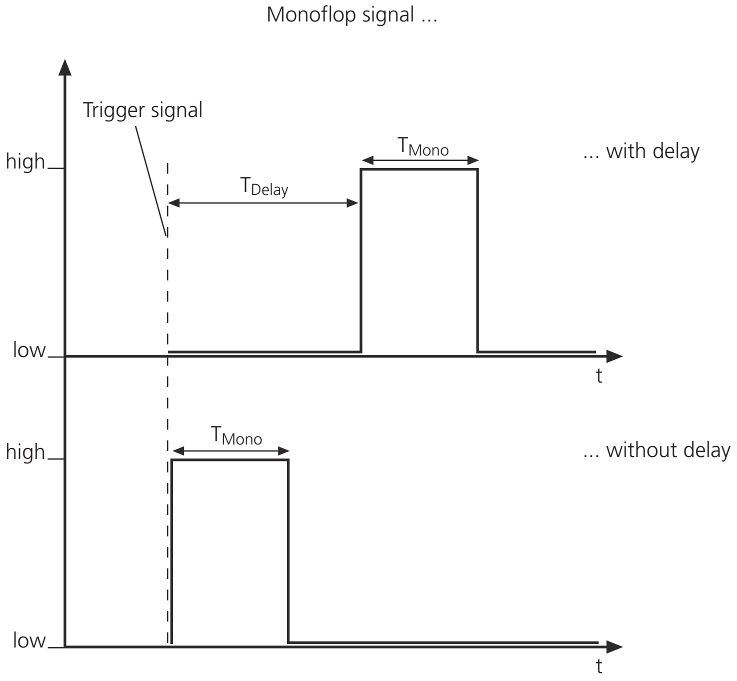
Purpose

To initialize delayed pulse generation with interrupt support.

Description

After initialization, channel operation must be triggered by using `ds4002_mono_start`, an internal or an external trigger. For further information, refer to Monoflop Signal Generation (DS4002 Features 📖).

The pulse width and the delay width may be updated by using the `ds4002_delayed_mono_int_update` function. If interrupt generation is enabled, on the rising or falling edge of the pulse an interrupt is generated. The pulse width is given by the `tm` parameter, the delay width is given by the `td` parameter. Depending on the number of active DS4002 channels, a minimum period and delay of 0.6 µs … 4 µs must be selected for `td` and `tm`. The period and the delay may be as long as 107 s. For details, refer to Monoflop Signal Generation (DS4002 Features 📖).



I/O mapping

For information on the I/O mapping, refer to Monoflop Signal Generation (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number.

> **Note**
>
> Only channel 1 and 2 are supporting interrupt generation. If interrupt generation is enabled and channel 3 … 8 are specified, the function exits with an error message.

td Specifies the pulse delay in seconds within the range 0.6e-6 … 107.0.

> **Note**
>
> If pulse generation is triggered by an external trigger, a constant delay of about 1 µs (with a jitter of 150 ns) occurs due to internal synchronization and processing times. Reduce the pulse delay appropriately for maximum accuracy. For details, refer to Triggering the Start of Signal Generation Externally (DS4002 Features 🕮).

tm Specifies the pulse width in seconds within the range 0.6 µs … 107.0 s.

intgen Enables the interrupt generation. The following symbols are predefined:

Symbol	Meaning
`DS4002_INT_NONE`	No interrupts
`DS4002_INT_RISING`	Interrupt on rising edge
`DS4002_INT_FALLING`	Interrupt on falling edge

Note

You cannot combine several different symbols.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds4002_delayed_mono_int_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.
-198	Error	ds4002_delayed_mono_int_init(0x??): Can't generate interrupts on channel ?!	The specified channel is unable to generate interrupts. Only channel 1 and 2 can generate interrupts in output mode.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

Monoflop Signal Generation (DS4002 Features 📖)

References

Base Address of the I/O Board 15
ds4002_delayed_mono_int_update 67
ds4002_init 19
DS4002_INT_CLEAR 16
DS4002_INT_STATUS 17
ds4002_mono_start 68

ds4002_delayed_mono_int_update

Syntax

```
void ds4002_delayed_mono_int_update (
   phs_addr_t base,
   long channel,
   dsfloat td,
   dsfloat tm)
```

Include file

`ds4002.h`

Purpose

To update the pulse and delay width of a channel.

Description

Updates will be effective for the next pulse. The pulse width is given by the `tm` parameter, the delay width is given by the `td` parameter.

> **Note**
>
> - The channel must have been initialized by using `ds4002_delayed_mono_int_init`.
> - Depending on the number of active DS4002 channels, a minimum period and delay of 0.6 µs to 4 µs must be selected. The period and the delay may be as long as 107 s.
>   For further information, refer to Monoflop Signal Generation (DS4002 Features 📖).

I/O mapping

For information on the I/O mapping, refer to Monoflop Signal Generation (DS4002 Features 📖).

Parameter

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

td Specifies the pulse delay in seconds within the range 0.6e-6 … 107.0.

> **Note**
>
> If pulse generation is triggered by an external trigger, a constant delay of about 1 µs (with a jitter of 150 ns) occurs due to internal synchronization and processing times. Reduce the pulse delay appropriately for maximum accuracy. For details, refer to Triggering the Start of Signal Generation Externally (DS4002 Features 📖).

tm Specifies the pulse width in seconds within the range 0.6 µs … 107.0 s.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

Monoflop Signal Generation (DS4002 Features 🕮)

References

Base Address of the I/O Board........ 15
ds4002_delayed_mono_int_init........ 64
ds4002_mono_start........ 68
Macros........ 15

ds4002_mono_start

Syntax

```
void ds4002_mono_start(
   phs_addr_t base,
   long channel)
```

Include file

`ds4002.h`

Purpose

To trigger the pulse generation for normal and delayed monoflop signals.

Description

The pulse generation of the specified channel is triggered. If the last pulse has not been completed it is terminated and a new pulse is started.

> **Note**
>
> The channel must have been initialized by using `ds4002_mono_init` or `ds4002_delayed_mono_int_init`.

I/O mapping

For information on the I/O mapping, refer to Monoflop Signal Generation (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the Monoflop Signal Generation Functions.......... 59

References

Base Address of the I/O Board.......... 15
ds4002_delayed_mono_int_init.......... 64
ds4002_mono_init.......... 61
Macros.......... 15

Arbitrary Signal Generation

Introduction

The timing I/O unit of the DS4002 allows you to flexibly generate complex digital pulse patterns on up to 8 channels.

Where to go from here

Information in this section

Information in other sections

Generation of Arbitrary Signals (DS4002 Features 🕮)
Using RTLib4002, you can also generate arbitrary pulse patterns.

Example of Using the Arbitrary Signal Generation Functions

Introduction

The following example demonstrates how to use the arbitrary signal generation functions of the DS4002. This example does not have any real background, but shall only show manual programming and the different update modes of the DS4002. You find the relevant files in `<RCP_HIL_InstallationPath>\Demos\Ds100<x>\IOBoards\DS4002\Cust_Out`. Use ControlDesk to load and start the application.

Tip

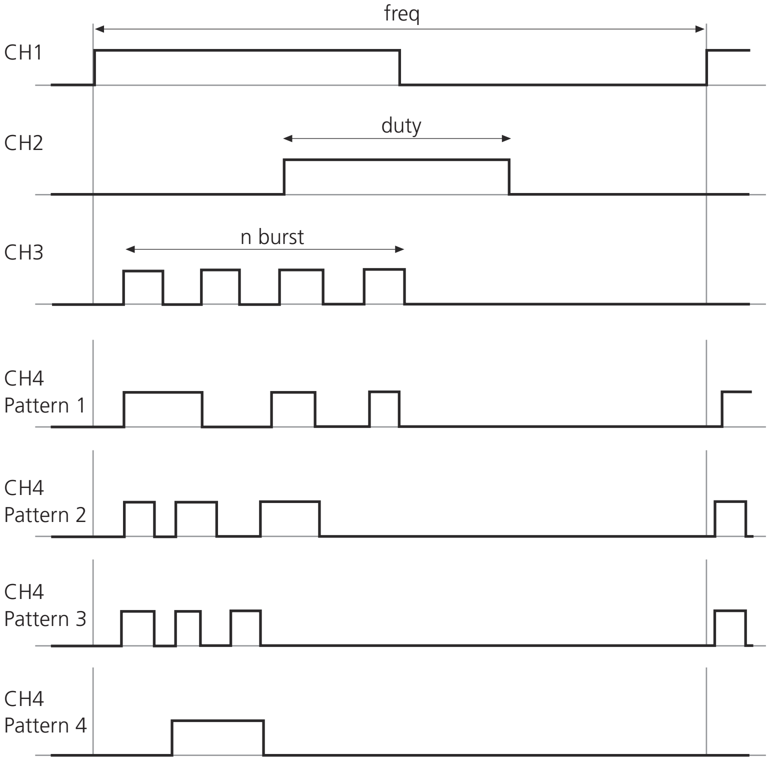
If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 4 are used for custom signal generation, which cannot be obtained by using the functions for generating standard digital pulse pattern such as PWM signals:

- Channel 1 acts as a main timer, which triggers channels 2 … 4, and generates a host interrupt (which increments a counter).
- Channel 2 delivers a PWM output signal.
- Channel 3 generates a burst signal with a variable amount of pulses.
- Channel 4 generates a pattern which can be selected from a list.

Channels 5 … 8 are not used in this example.



All channels are updated in an interrupt service routine every 1 ms.

You have to connect the channels 1 … 4 to an oscilloscope.

> **Note**
>
> For controlling of the signal generating channels, use the `custout_4002_hc.cdp` project with ControlDesk. The layout of the experiment provides a radio button instrument to select the active pattern of channel 4.

```
#include "brtenv.h"                    /* basic real time environment */
#include "ds4002.h"                    /* DS4002 constants and macros *
/
/************************************************************************
  global variables
************************************************************************/
dsfloat freq     = 1.0e3;
dsfloat duty     = 0.4;
long    nburst   = 4;
dsfloat tburstl  = 6.0e-6;
dsfloat tbursth  = 8.0e-6;
long    npattern = 0;
long    intcount = 0;
int addr1[2];                /* list of update addresses for channel 1 */
int addr2[3];                /* list of update addresses for channel 2 */
int addr3[3];                /* list of update addresses for channel 3 */
int addr4[1];                /* list of update addresses for channel 4 */
int pattern[4];   /* entry points for different patterns for channel 4 */
```

```
/********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long dl,dh;
  ts_timestamp_type ts;
  if (freq > 14e3) freq = 14e3;          /* limit freq to 14khz */
           /* because pattern 0 and 1 of channel 3 need 70usecs */
  /* update channel 1 */
  dl = DS4002_DELAY(0.5/freq);
  ds4002_update_state(DS4002_1_BASE, 1, addr1[0],
    dl,                                       /* after delay 0.5/freq */
    DS4002_HIGH,                                   /* set output high */
                                                      /* please note:
                 update of trigger or interrupt data is not possible! */
    DS4002_CONTINUE,                        /* continue with next state */
    0);                                /* no loop counter or jump value */
  ds4002_update_state(DS4002_1_BASE, 1, addr1[1],
    dl,                                       /* after delay 0.5/freq */
    DS4002_LOW,                                     /* set output low */
    DS4002_GOTO,                  /* goto entry point (= first state) */
    0);                                /* no loop counter or jump value */
  DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_IMMEDIATE, 1);
                      /* advance swinging buffer immediately and update
                                               currently used delays */
  /* update channel 2 */
  dh = DS4002_DELAY(duty / freq);
  dl = DS4002_DELAY(0.5 * (1 - duty) / freq);
  if (dh < 5)
  {                               /* duty too small, output stays low */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[0],
      DS4002_WAIT,                            /* wait for trigger event */
      DS4002_LOW,               /* after trigger event set output low */
      DS4002_CONTINUE,                    /* continue with next state */
      0);                              /* no loop counter or jump value */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[1],
      10,                                           /* after 10 Ticks */
      DS4002_LOW,                                   /* set output low */
      DS4002_CONTINUE,                    /* continue with next state */
      0);                              /* no loop counter or jump value */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[2],
      10,                                           /* after 10 Ticks */
      DS4002_LOW,                                   /* set output low */
      DS4002_GOTO,                /* goto entry point (= first state) */
      0);                              /* no loop counter or jump value */
  }
  else if (dl < 10)
  {                                /* duty too big, output stays high */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[0],
      DS4002_WAIT,                            /* wait for trigger event */
      DS4002_HIGH,             /* after trigger event set output high */
      DS4002_CONTINUE,                    /* continue with next state */
      0);                              /* no loop counter or jump value */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[1],
      10,                                           /* after 10 Ticks */
      DS4002_HIGH,                                 /* set output high */
      DS4002_CONTINUE,                    /* continue with next state */
      0);                              /* no loop counter or jump value */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[2],
      10,                                           /* after 10 Ticks */
      DS4002_HIGH,                                 /* set output high */
      DS4002_GOTO,                /* goto entry point (= first state) */
      0);                            /* no loop counter or jump value */
  }
  else
  {
    ds4002_update_state(DS4002_1_BASE, 2, addr2[0],
      DS4002_WAIT,                          /* wait for trigger event */
      DS4002_LOW,               /* after trigger event set output low */
      DS4002_CONTINUE,                    /* continue with next state */
      0);                            /* no loop counter or jump value */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[1],
      dl,                                     /* after 1/2 low period */
      DS4002_HIGH,                                 /* set output high */
      DS4002_CONTINUE,                    /* continue with next state */
      0);                            /* no loop counter or jump value */
    ds4002_update_state(DS4002_1_BASE, 2, addr2[2],
      dh,                                        /* after high period */
      DS4002_LOW,                                   /* set output low */
      DS4002_GOTO,                /* goto entry point (= first state) */
      0);                            /* no loop counter or jump value */
  }
  DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_NEWDATA, 2);
            /* advance swinging buffer to be used with the next delay */
  /* update channel 3 */
                                                       /* range check */
  if ((tburstl + tbursth + 5e-6) > (1/freq))
  {                    /* limit to one pulse with reduced pulse width */
    tburstl = 0.5 * (1/freq - 5e-6);
    tbursth = tburstl;
    nburst = 1;
  }
  if ((nburst * (tburstl + tbursth) + 5e-6) > (1/freq))
  {                                         /* limit number of pulses */
    nburst = (long)(((1/freq) - 5e-6) / (tburstl + tbursth));
  }
  ds4002_update_state(DS4002_1_BASE, 3, addr3[0],
    DS4002_DELAY(2e-6),                          /* after 2 microsecs */
    DS4002_LOW,                                     /* set output low */
    DS4002_LOADCOUNTER,             /* continue with next state and */
    nburst);                                     /* load loop counter */
  ds4002_update_state(DS4002_1_BASE, 3, addr3[1],
    DS4002_DELAY(tburstl),                     /* after delay tburstl */
    DS4002_HIGH,                                   /* set output high */
    DS4002_CONTINUE,                      /* continue with next state */
    0);                              /* no loop counter or jump value */
  ds4002_update_state(DS4002_1_BASE, 3, addr3[2],
    DS4002_DELAY(tbursth),                     /* after delay tbursth */
    DS4002_LOW,                                     /* set output low */
    DS4002_REPEAT,            /* decrement loop counter, if not zero,
               goto local entry label. Else, continue with next state */
    0);                              /* no loop counter or jump value */
  DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_BLOCKDATA, 3);
             /* advance swinging buffer to be used after the execution
                                                    of a GOTO command */
  /* update channel 4 */
  ds4002_update_state(DS4002_1_BASE, 4, addr4[0],
    DS4002_DELAY(2e-6),                    /* after delay 2 microsecs */
    DS4002_LOW,                                     /* set output low */
    DS4002_JUMP,                               /* jump to first state */
    pattern[npattern]);                        /* of selected pattern */
  DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_BLOCKDATA, 4);
             /* advance swinging buffer to be used after the execution
                                                    of a GOTO command */
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
void channel1_intserv()
{
  intcount++;
}
/*********************************************************************
  main
*********************************************************************/
void main()
{
  init();                          /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);           /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  install_phs_int_vector(       /* initialize interrupt controllers */
     DS4002_1_BASE,                             /* board base address */
     1,                  /* slave ICU input
                          0 = ILEN interrupt in input mode
                             (check INT register for channel numbers)
                          1 = channel 1 in output mode
                          2 = channel 2 in output mode               */
     channel1_intserv );               /* address of service routine */
  /* ch1: main clock generator, triggers ch2-4 */
  ds4002_output_init();                 /* prepare program variables */
  ds4002_define_entry();              /* entry point = program start */
  addr1[0] = ds4002_define_state(
     DS4002_DELAY(0.5/freq),                /* after delay 0.5/freq */
     DS4002_HIGH,                                /* set output high */
     0,                               /* do not trigger or interrupt */
     DS4002_CONTINUE,                    /* continue with next state */
     0);                            /* no loop counter or jump value */
  addr1[1] = ds4002_define_state(
     DS4002_DELAY(0.5/freq),                /* after delay 0.5/freq */
     DS4002_LOW,                                  /* set output low */
     DS4002_MASK(2)+DS4002_MASK(3)+DS4002_MASK(4)+DS4002_INTERRUPT,
                /* trigger channels 2 to 4, generate host interrupt */
     DS4002_GOTO,               /* goto entry point (= first state) */
     0);                            /* no loop counter or jump value */
  ds4002_load_states(DS4002_1_BASE, 1);
                                  /* download program for channel 1 */
  /* ch2: pwm output */
  ds4002_output_init();                 /* prepare program variables */
  ds4002_define_entry();              /* entry point = program start */
  addr2[0] = ds4002_define_state(
     DS4002_WAIT,                         /* wait for trigger event */
     DS4002_LOW,              /* after trigger event set output low */
     0,                               /* do not trigger or interrupt */
     DS4002_CONTINUE,                    /* continue with next state */
     0);                            /* no loop counter or jump value */
```

```
addr2[1] = ds4002_define_state(
   DS4002_DELAY(0.5/freq * (1-duty)),     /* after 1/2 low period */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
addr2[2] = ds4002_define_state(
   DS4002_DELAY(1/freq * duty),              /* after high period */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_GOTO,               /* goto entry point (= first state) */
   0);                           /* no loop counter or jump value */
ds4002_load_states(DS4002_1_BASE, 2);
                                /* download program for channel 2 */
/* ch3: burst output */
ds4002_output_init();                /* prepare program variables */
ds4002_define_entry();             /* entry point = program start */
ds4002_define_state(
   DS4002_WAIT,                         /* wait for trigger event */
   DS4002_LOW,              /* after trigger event set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
addr3[0] = ds4002_define_state(
   DS4002_DELAY(2e-6),                       /* after 2 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_LOADCOUNTER,            /* continue with next state and */
   nburst);                                  /* load loop counter */
   /* LOADCOUNTER sets a local entry label for the REPEAT command */
addr3[1] = ds4002_define_state(
   DS4002_DELAY(tburstl),                  /* after delay tburstl */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
addr3[2] = ds4002_define_state(
   DS4002_DELAY(tbursth),                  /* after delay tbursth */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_REPEAT,/* decrement loop counter. If not zero, goto local
                      entry label. Else, continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(2e-6),                       /* after 2 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_GOTO,               /* goto entry point (= first state) */
   0);                           /* no loop counter or jump value */
ds4002_load_states(DS4002_1_BASE, 3);
                                /* download program for channel 3 */
/* ch4: variable patterns */
ds4002_output_init();                /* prepare program variables */
ds4002_define_entry();             /* entry point = program start */
ds4002_define_state(
   DS4002_WAIT,                         /* wait for trigger event */
   DS4002_LOW,              /* after trigger event set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
```

```
addr4[0] = ds4002_define_state(
   DS4002_DELAY(2e-6),                 /* after delay 2 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_JUMP,                                  /* jump to state */
   3);                                /* at address 3 (pattern 0) */
             /* note: due to the jump, this state needs two words */
 /* we have to guess the value for pattern[0] here, because it is */
 /* not yet available. The first two states need 1 + 2 words, so  */
 /* pattern 0 will start at address 3.                            */
/* pattern 0 */
pattern[0] = ds4002_define_state(
   DS4002_DELAY(20e-6),                     /* after 20 microsecs */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(20e-6),                     /* after 20 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(10e-6),                     /* after 10 microsecs */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(10e-6),                     /* after 10 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(5e-6),                       /* after 5 microsecs */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(5e-6),                       /* after 5 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_GOTO,               /* goto entry point (= first state) */
   0);                           /* no loop counter or jump value */
/* pattern 1 */
pattern[1] = ds4002_define_state(
   DS4002_DELAY(5e-6),                       /* after 5 microsecs */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_DELAY(5e-6),                       /* after 5 microsecs */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(10e-6),                     /* after 10 microsecs */
    DS4002_HIGH,                                /* set output high */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(10e-6),                     /* after 10 microsecs */
    DS4002_LOW,                                  /* set output low */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(20e-6),                     /* after 20 microsecs */
    DS4002_HIGH,                                /* set output high */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(20e-6),                     /* after 20 microsecs */
    DS4002_LOW,                                  /* set output low */
    0,                              /* do not trigger or interrupt */
    DS4002_GOTO,               /* goto entry point (= first state) */
    0);                           /* no loop counter or jump value */
 /* pattern 2 */
 pattern[2] = ds4002_define_state(
    DS4002_DELAY(5e-6),                       /* after 5 microsecs */
    DS4002_HIGH,                                /* set output high */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
ds4002_define_state(
    DS4002_DELAY(5e-6),                       /* after 5 microsecs */
    DS4002_LOW,                                  /* set output low */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(5e-6),                       /* after 5 microsecs */
    DS4002_HIGH,                                /* set output high */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(5e-6),                       /* after 5 microsecs */
    DS4002_LOW,                                  /* set output low */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(5e-6),                       /* after 5 microsecs */
    DS4002_HIGH,                                /* set output high */
    0,                              /* do not trigger or interrupt */
    DS4002_CONTINUE,                   /* continue with next state */
    0);                           /* no loop counter or jump value */
 ds4002_define_state(
    DS4002_DELAY(5e-6),                       /* after 5 microsecs */
    DS4002_LOW,                                  /* set output low */
    0,                              /* do not trigger or interrupt */
    DS4002_GOTO,               /* goto entry point (= first state) */
    0);                           /* no loop counter or jump value */
```

```
  /* pattern 3 */
  pattern[3] = ds4002_define_state(
     DS4002_DELAY(20e-6),                     /* after 20 microsecs */
     DS4002_HIGH,                                /* set output high */
     0,                              /* do not trigger or interrupt */
     DS4002_CONTINUE,                   /* continue with next state */
     0);                           /* no loop counter or jump value */
  ds4002_define_state(
     DS4002_DELAY(20e-6),                     /* after 20 microsecs */
     DS4002_LOW,                                  /* set output low */
     0,                              /* do not trigger or interrupt */
     DS4002_GOTO,               /* goto entry point (= first state) */
     0);                           /* no loop counter or jump value */
  ds4002_load_states(DS4002_1_BASE, 4);
                                  /* download program for channel 4 */
  ds4002_start_channels(DS4002_1_BASE,      /* start channel 1 to 4 */
   DS4002_MASK(1) + DS4002_MASK(2) + DS4002_MASK(3) + DS4002_MASK(4) );
  RTLIB_SRT_START(0.001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples

Example of Implementing Arbitrary Signal Generation Code as S-Function................................ 79

Example of Implementing Arbitrary Signal Generation Code as S-Function

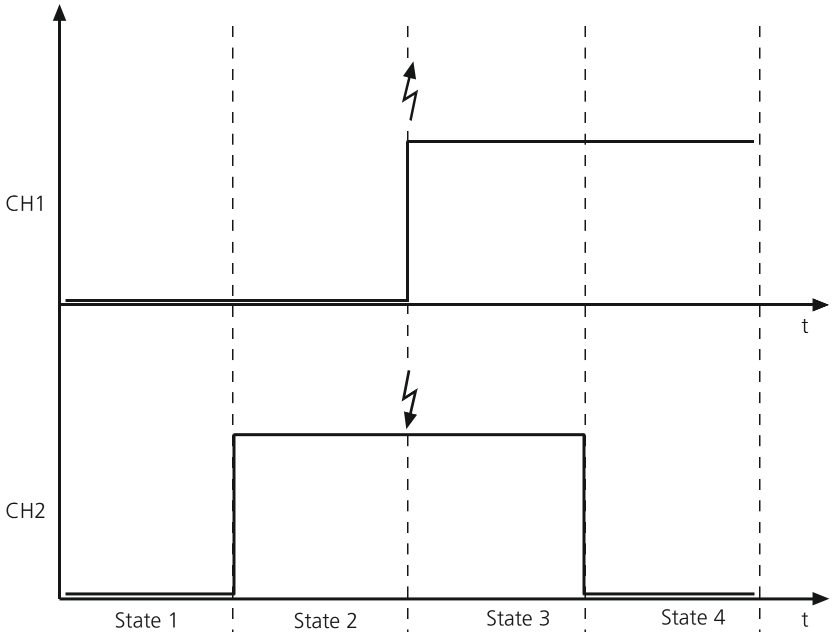
Introduction

The following example demonstrates how to use the arbitrary signal generation functions of the DS4002. It is an emulation program for an incremental encoder. Incremental encoders provide the two encoder signals PHI0 and PHI90 and the index signal IDX. The encoder signal pair PHI0 <-> PHI90 has a phase shift of 90°.

Description

Channels 1 and 2 are used for arbitrary signal generation, which cannot be obtained by using the functions for generating standard digital pulse pattern such as PWM signals:

- Channel 1 acts as a main timer that triggers channel 2.
- Channel 2 waits for the trigger of channel 1.



```
#define S_FUNCTION_NAME ds4002_enc_emu_sfcn
#include "tmwtypes.h"
#include "simstruc.h"
#ifdef MATLAB_MEX_FILE
  static char *RCSfile = "$RCSfile: ds4002_enc_emu_sfcn.c $";
  static char *RCSrev  = "$Revision: 1.0 $";
  static char *RCSdate = "$Date: 1999/03/11 08:33:24 $";
#endif
#ifndef MATLAB_MEX_FILE
   #include <brtenv.h>
   #include <ds4002.h>
   #include <rtierrhndl.h>
#endif
#define SFCN_NUM_PARAM_ERROR 199
#define NUM_INPUTS       1
#define NUM_OUTPUTS      0
#define NUM_PARAM        5
#define BOARD_NUMBER    (int_T) (mxGetPr(ssGetArg(S,0))[0])
#define LINES            (real_T)(mxGetPr(ssGetArg(S,1))[0])
#define RPM_INIT         (real_T)(mxGetPr(ssGetArg(S,2))[0])
#define CHANNEL_1    (int_T) (mxGetPr(ssGetArg(S,3))[0])
#define CHANNEL_2    (int_T) (mxGetPr(ssGetArg(S,4))[0])
#define BOARD_BASE       *IWork
long addr1[4];           // Update addresses for PHI0
long addr2[3];           // Update addresses for PHI90
static void mdlInitializeSizes(SimStruct *S)
{
  ssSetNumSFcnParams(S, NUM_PARAM);
  if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
  {
    #ifndef MATLAB_MEX_FILE
      signal_error(SFCN_NUM_PARAM_ERROR);
    #endif
    return;
  }
  ssSetNumContStates(    S, 0);
  ssSetNumDiscStates(    S, 0);
  ssSetNumInputs(        S, NUM_INPUTS);
  ssSetNumOutputs(       S, NUM_OUTPUTS);
  ssSetDirectFeedThrough(S, 0);
  ssSetNumSampleTimes(   S, 1);
  ssSetNumRWork(         S, 0);
  ssSetNumIWork(         S, 1);
  ssSetNumPWork(         S, 0);
  ssSetNumModes(         S, 0);
  ssSetNumNonsampledZCs( S, 0);
  ssSetOptions(          S, 0);
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
}
static void mdlInitializeConditions(real_T *x0, SimStruct *S)
{
#ifndef MATLAB_MEX_FILE
   long phi0_base;                                        /* base delay */
   int   *IWork = ssGetIWork(S);
   long  board_index = (long) BOARD_NUMBER;
   switch(board_index)                            /* select board index */
   {
      case 1   : *IWork = DS4002_1_BASE;
                 break;
      case 2   : *IWork = DS4002_2_BASE;
                 break;
      default  : return;
                 break;
   }
   phi0_base = DS4002_DELAY(60/(RPM_INIT*LINES*4));   /* calculate delay */
   ds4002_init(BOARD_BASE);                   /* initialize DS4002 Board */
   ds4002_output_init();                    /* prepare program variables */
  ds4002_define_entry ();                 /* entry point = program start */
   addr1[0] = ds4002_define_state(phi0_base,     /* Init Channel1 State1 */
                                  DS4002_LOW,
                                  0,
                                  DS4002_CONTINUE,
                                  0);
   addr1[1] = ds4002_define_state(phi0_base,     /* Init Channel1 State2 */
                                  DS4002_HIGH,
                                  DS4002_MASK(CHANNEL_2), /* Trigger PHI90 */
                                  DS4002_CONTINUE,
                                  0);
   addr1[2] = ds4002_define_state(phi0_base,     /* Init Channel1 State3 */
                                  DS4002_HIGH,
                                  0,
                                  DS4002_CONTINUE,
                                  0);
   addr1[3] = ds4002_define_state(phi0_base,     /* Init Channel1 State4 */
                                  DS4002_LOW,
                                  0,
                                  DS4002_GOTO,
                                  0);
   ds4002_load_states(BOARD_BASE, CHANNEL_1);      /* States to Channel1 */
   ds4002_output_init();                    /* prepare program variables */
   ds4002_define_entry();                 /* entry point = program start */
   addr2[0] = ds4002_define_state(phi0_base,     /* Init Channel2 State1 */
                                  DS4002_HIGH,
                                  0,
                                  DS4002_CONTINUE,
                                  0);
   ds4002_define_state(DS4002_WAIT, /* Init Channel2 State2 wait trigger */
                       DS4002_HIGH,
                       0,
                       DS4002_CONTINUE,
                       0);
   addr2[1] = ds4002_define_state(phi0_base,     /* Init Channel2 State3 */
                                  DS4002_LOW,
                                  0,
                                  DS4002_CONTINUE,
                                  0);
   addr2[2] = ds4002_define_state(phi0_base,     /* Init Channel2 State4 */
                                  DS4002_LOW,
                                  0,
                                  DS4002_GOTO,
                                  0);
   ds4002_load_states(BOARD_BASE, CHANNEL_2);      /* States to Channel2 */
   ds4002_start_channels(BOARD_BASE,                  /* Start Channels */
                         DS4002_MASK(CHANNEL_1)|
                         DS4002_MASK(CHANNEL_2));
#endif
}
static void mdlOutputs(real_T *y, const real_T *x, const real_T *u,
                       SimStruct *S, int_T tid)
{
  #ifndef MATLAB_MEX_FILE
    long phi0;
    float delay;
    int  *IWork = ssGetIWork(S);
    delay = 60/(abs(u[0])*LINES*4);
    delay = (delay < 107.374) ? delay : 107.374;
    phi0 = DS4002_DELAY(delay);
    if(u[0])>=0)                                              /* rpm >= 0 */
    {
       ds4002_update_state(BOARD_BASE, /* Update delay time 1 of channel1 */
                           CHANNEL_1,
                           addr1[0],
                           phi0,
                           DS4002_LOW,
                           DS4002_CONTINUE,
                           0);
       ds4002_update_state(BOARD_BASE, /* Update delay time 2 of channel1 */
                           CHANNEL_1,
                           addr1[1],
                           phi0,
                           DS4002_HIGH,
                           DS4002_CONTINUE,
                           0);
       ds4002_update_state(BOARD_BASE, /* Update delay time 3 of channel1 */
                           CHANNEL_1,
                           addr1[2],
                           phi0,
                           DS4002_HIGH,
                           DS4002_CONTINUE,
                           0);
       ds4002_update_state(BOARD_BASE, /* Update delay time 4 of channel1 */
                           CHANNEL_1,
                           addr1[3],
                           phi0,
                           DS4002_LOW,
                           DS4002_GOTO,
                           0);
    }
    else                                                       /* rpm < 0 */
    {
       ds4002_update_state(BOARD_BASE, /* Update delay time 1 of channel1 */
                           CHANNEL_1,
                           addr1[0],
                           phi0,
                           DS4002_HIGH,
                           DS4002_CONTINUE,
                           0);
       ds4002_update_state(BOARD_BASE, /* Update delay time 2 of channel1 */
                           CHANNEL_1,
                           addr1[1],
                           phi0,
                           DS4002_LOW,
                           DS4002_CONTINUE,
                           0);
       ds4002_update_state(BOARD_BASE, /* Update delay time 3 of channel1 */
                           CHANNEL_1,
                           addr1[2],
                           phi0,
                           DS4002_LOW,
                           DS4002_CONTINUE,
                           0);
       ds4002_update_state(BOARD_BASE, /* Update delay time 4 of channel1 */
                           CHANNEL_1,
                           addr1[3],
                           phi0,
                           DS4002_HIGH,
                           DS4002_GOTO,
                           0);
    }
    ds4002_update_state(BOARD_BASE,  /* Update delay time 1 of channel2 */
                        CHANNEL_2,
                        addr2[0],
                        phi0,
                        DS4002_HIGH,
                        DS4002_CONTINUE,
                        0);
    ds4002_update_state(BOARD_BASE,  /* Update delay time 3 of channel2 */
                        CHANNEL_2,
                        addr2[1],
                        phi0,
                        DS4002_LOW,
                        DS4002_CONTINUE,
                        0);
    ds4002_update_state(BOARD_BASE,  /* Update delay time 4 of channel2 */
                        CHANNEL_2,
                        addr2[2],
                        phi0,
                        DS4002_LOW,
                        DS4002_GOTO,
                        0);
```

```
    DS4002_EXEC_CMD(BOARD_BASE, DS4002_CMD_IMMEDIATE, CHANNEL_1);
    DS4002_EXEC_CMD(BOARD_BASE, DS4002_CMD_IMMEDIATE, CHANNEL_2);
  #endif
}
static void mdlUpdate(real_T *x, const real_T *u, SimStruct *S, int_T tid)
{
}
static void mdlDerivatives(real_T *dx, const real_T *x, const real_T *u,
                           SimStruct *S, int_T tid)
{
}
static void mdlTerminate(SimStruct *S)
{
}
#ifdef   MATLAB_MEX_FILE     /* Is this file being compiled as a MEX-file? */
#include "simulink.c"                      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"              /* Code generation registration function */
#endif
```

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions.. 71

References

ds4002_define_entry.. 89
ds4002_define_state.. 85
DS4002_DELAY.. 97
DS4002_EXEC_CMD.. 98
ds4002_init.. 19
ds4002_load_states.. 90
DS4002_MASK.. 101
ds4002_output_init.. 84
ds4002_start_channels... 91
ds4002_update_state.. 92

ds4002_output_init

Syntax

```
void ds4002_output_init(void)
```

Include file

`ds4002.h`

Purpose

To initialize a state machine code for programming an arbitrary signal generation.

Description

A reserved temporary memory buffer of 256 words is cleared which will contain the new state machine code. The address of the first state in the new code is set as default entry point. This entry point address can be changed by using the `ds4002_define_entry` function.

I/O mapping

For information on the I/O mapping, refer to Generation of Arbitrary Signals (DS4002 Features).

Return value

None

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions.. 71

References

ds4002_define_entry.. 89

ds4002_define_state

Syntax

```
long ds4002_define_state(
   long delay,
   long level,
   long trigger,
   long instr,
   long count)
```

Include file

`ds4002.h`

Purpose

To define a single state within the state machine code.

Description

Each state consists of a delay, an output level and a command for the program flow.

> **Note**
>
> Before you can use the flexible signal generation, you must download the state machine code by using the `ds4002_load_states` function. If you have specified the signal generation for all required channels, you can start the execution with the `ds4002_start_channels` function.

I/O mapping

For information on the I/O mapping, refer to Generation of Arbitrary Signals (DS4002 Features 📖).

Parameters

delay Specifies the delay value in time base tics (1 tic = 200 ns) after that actions (for example, change output level, trigger other channels, generate interrupts) have to be executed. To calculate the time base tics from a specified time value in seconds or a specified angle in degree, use the `DS4002_DELAY`, `DS4002_ANGLE` or `DS4002_ANGLE2` macro, corresponding to the specified mode (time-based or angle-based).

Conversion Macro	Delay Value Range
Without conversion (tics)	0 … 0x1FFFFFFF
`DS4002_DELAY()`	0.0 … 107.374 s
`DS4002_ANGLE()`	0.0 … 179.99°
`DS4002_ANGLE2()`	0.0 … 359.99°

Note

- The minimum delay value depends on the number of active channels and on the channel priority. You may use shorter delays, even a value of zero is possible. In this case, the channels are serviced as fast as possible. Some edges may be delayed, but the next edges will occur at the correct time again (delay errors are compensated with the next delay and not accumulated). For further information, refer to Defining and Specifying States (DS4002 Features 🕮).
- If you have specified delay values greater than the maximum mentioned above, there will be unpredictable results.
- If the channel shall wait for a trigger from channels 1 or 2 or for an external trigger from TRIGA or TRIGB, use the `DS4002_WAIT` constant as delay. In this case all actions are performed after the trigger event occurred. If pulse generation is triggered by an external trigger, a constant delay of about 1 µs (with a jitter of 150 ns) occurs due to internal synchronization and processing times. For details, refer to Triggering the Start of Signal Generation Externally (DS4002 Features 🕮).

level Specifies the level which appears at the channel output after the delay has expired. The following symbols are predefined:

Symbol	Description
`DS4002_HIGH`	High level
`DS4002_LOW`	Low level

trigger Specifies a trigger and interrupt instruction. Use the following predefined macros and symbols:

Value	Meaning
`DS4002_MASK(channel)`	Specifies a trigger request to one or more DS4002 channels. To specify more trigger request, You can use `DS4002_MASK()` several times, for example, `DS4002_MASK(1) | DS4002_MASK(3)`
`DS4002_INTERRUPT`	Specifies an PHS-Bus interrupt request.
0	Specifies no request for trigger or interrupt.

Note

Only channels 1 and 2 can be used for trigger and interrupt instructions.

instr Specifies a program flow instruction. States using the `DS4002_LOADCOUNTER` or the `DS4002_JUMP` instruction need 2 words in the state machine code, all other states only need 1 word. A maximum of 256 program words can be used.

The following instructions are available:

Flow Instruction	Meaning
`DS4002_CONTINUE`	Continues with next state.
`DS4002_GOTO`	Continues with the state, which was defined after using `ds4002_define_entry.`
`DS4002_JUMP`	Continue with the state at the given address. This command may not be combined with the `DS4002_WAIT` directive, and it may not be used within a loop construct, i.e. between a `DS4002_LOADCOUNTER` and a `DS4002_REPEAT` instruction!
`DS4002_LOADCOUNTER`	Continues with next state, save address of next state as a local label, and load the loop counter with the count. This command may not be combined with the `DS4002_WAIT` directive!
`DS4002_REPEAT`	Decrement the loop counter; if zero, continue with next state, if not zero, continue with the state following the `DS4002_LOADCOUNTER` state.

For further information, refer to Defining and Specifying States (DS4002 Features 📖).

count Specifies the value for the given flow instruction. The following values can be used:

Flow Instruction	Value	Meaning
`DS4002_LOADCOUNTER`	1 … 256	Specifies the loop counter.
`DS4002_JUMP`	0 … 255	Specifies the jump address, which is returned by the `ds4002_define_state` function.

`DS4002_CONTINUE`
`DS4002_GOTO`
`DS4002_REPEAT`
	0	No value required. Must be 0.

Return value

Returns the address of the state which is defined. The address is used for the `ds4002_update_state` function, or for the jump instruction.

Example

This example shows how to use the function:

```
upd_addr1 = ds4002_define_state
   (DS4002_DELAY(0.001), DS4002_HIGH, DS4002_MASK(3) +
    DS4002_INTERRUPT, DS4002_LOADCOUNTER, 10);
```

After 1 ms set output to high level, trigger channel 3, generate a host interrupt, load the loop counter with 10 and continue at the next state.

Save the next state address as a local entry point for the `DS4002_REPEAT` instruction. Return the current state address for use with the `ds4002_update_state` function.

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions.. 71

References

ds4002_define_entry.. 89
ds4002_load_states.. 90
ds4002_start_channels.. 91
ds4002_update_state.. 92
Macros.. 15

ds4002_define_entry

Syntax

```
void ds4002_define_entry(void)
```

Include file

`ds4002.h`

Purpose

To define an entry point in the state machine code.

Description

The entry point address specifies the next defined state as target state for the following `DS4002_GOTO` instruction (see `ds4002_define_state` on page 85). If you do not use this function, a `DS4002_GOTO` instruction jumps to the first state of the state machine code. This is the default entry point address initialized by the `ds4002_output_init` function.

Note

- This function may only be used once per state machine code, because a further `ds4002_define_entry` call will overwrite the first definition of the entry point.
- Before you can generate signals, you must download the state machine code by using the `ds4002_load_states` function. If you have specified the signal generation for all required channels, you can start the execution with the `ds4002_start_channels` function.

I/O mapping

For information on the I/O mapping, refer to Generation of Arbitrary Signals (DS4002 Features 🕮).

Return value

None

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions........ 71

References

ds4002_define_state........ 85
ds4002_load_states........ 90
ds4002_output_init........ 84
ds4002_start_channels........ 91
Macros........ 15

ds4002_load_states

Syntax

```
void ds4002_load_states(
   phs_addr_t base,
   long channel)
```

Include file

`ds4002.h`

Purpose

To copy the state machine code to a DS4002 channel.

Description

Using this function, the defined state machine code is downloaded to the specified channel of the DS4002. If you want to use the same state machine code for more than one channel, you must call this function for each required channel.

> **Note**
>
> The channel operation must be started by using the `ds4002_start_channels` function.

I/O mapping

For information on the I/O mapping, refer to Generation of Arbitrary Signals (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

Example This example shows how to load the state machine code to channel 2:

```
…
ds4002_load_states (DS4002_1_BASE, 2);
```

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions.......... 71

References

Base Address of the I/O Board.......... 15
ds4002_start_channels.......... 91

ds4002_start_channels

Syntax

```
void ds4002_start_channels(
   phs_addr_t base,
   long mask)
```

Include file

`ds4002.h`

Purpose

To enable the signal generation on the specified channels.

Description

After you have loaded the state machine codes for the required channels, you can start the channels with this function.

I/O mapping

For information on the I/O mapping, refer to Generation of Arbitrary Signals (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

mask Specifies the channels for signal generation. Use the `DS4002_MASK(channel)` macro to set this value.

Return value

None

Example

This example shows how to enable signal generation on channels 1 and 3:

```
ds4002_start_channels (DS4002_1_BASE,
DS4002_MASK(1) + DS4002_MASK(3));
```

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions.. 71

References

Base Address of the I/O Board.. 15
Macros... 15

ds4002_update_state

Syntax

```
void ds4002_update_state(
   phs_addr_t base,
   long channel,
   long state,
   long delay,
   long level,
   long instr,
   long count)
```

Include file

`ds4002.h`

Purpose

To update a single state within a DS4002 output program.

Description

This function is used to update a single state within a DS4002 output program. The `delay`, `level`, `instr` and `count` parameters are the same as used for defining a state with `ds4002_define_state`.

Note

- Change of trigger and interrupt instructions within a state is not possible.
- The `instr` parameter must be the same as used in `ds4002_define_state`. It is not possible to change the program flow during an update. Also, do not change from a delay to the `DS4002_WAIT` constant or vice versa.

Reasonable updates would be:

- the change of a delay value,
- the change of an output level,
- the change of a jump address, or/and
- the change of a loop counter value.

Do not try to read back a state, modify a part of it and write it back. The state you read is not the most actual one, but comes from an update some time ago.

The update is performed only in the swinging buffer section which is visible for the host. In order to advance the swinging buffer controller and making the changes effective, the `DS4002_EXEC_CMD` macro must be executed with a special command. Furthermore, there are several update commands for different update modes.

For further information about the swinging buffer update, refer to Swinging Buffer Principle (DS4002 Features 🕮), for information on the update modes, refer to Updating State Parameters (DS4002 Features 🕮).

The modified states will be effective after execution of the specified update command by using the `DS4002_EXEC_CMD` macro.

Note

Do not update different parameters of the same channel at different points in your host software. For example, avoid a construct like this:

```
…
ds4002_update_state(base, channel1, state1, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_IMMEDIATE, channel1);
…
ds4002_update_state(base, channel1, state2, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_IMMEDIATE, channel1);
…
```

The first update will affect state1, as planned. The second one however will update state2, but also reset state1 to its previous parameters. Updates only affect one of three swinging buffers, so be sure to always update all parameters that can change (even if they remain constant during this update).

For further information about the swinging buffer update, refer to Swinging Buffer Principle (DS4002 Features 🕮). For information on the update modes, refer to Updating State Parameters (DS4002 Features 🕮).

I/O mapping

For information on the I/O mapping, refer to Generation of Arbitrary Signals (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

state Specifies the address of the state to be updated. The value for this parameter can be obtained as return value from `ds4002_define_state`.

delay Specifies the delay value in time base tics (1 tic = 200 ns) after that actions (for example, change output level, trigger other channels, generate interrupts) have to be executed. To calculate the time base tics from a specified time value in seconds or a specified angle in degree, use the `DS4002_DELAY`, `DS4002_ANGLE` or `DS4002_ANGLE2` macro, corresponding to the specified mode (time-based or angle-based).

Conversion Macro	Delay Value Range
Without conversion (tics)	0 … 0x1FFFFFFF
`DS4002_DELAY()`	0.0 … 107.374 s
`DS4002_ANGLE()`	0.0 … 179.99°
`DS4002_ANGLE2()`	0.0 … 359.99°

Note

- The minimum delay value depends on the number of active channels and on the channel priority. You may use shorter delays, even a value of zero is possible. In this case, the channels are serviced as fast as possible. Some edges may be delayed, but the next edges will occur at the correct time again (delay errors are compensated with the next delay and not accumulated). For further information, refer to Defining and Specifying States (DS4002 Features 🕮).
- If you have specified delay values greater than the maximum mentioned above, there will be unpredictable results.
- If the channel shall wait for a trigger from channels 1 or 2 or for an external trigger from TRIGA or TRIGB, use the `DS4002_WAIT` constant as delay. In this case all actions are performed after the trigger event occurred. If pulse generation is triggered by an external trigger, a constant delay of about 1 µs (with a jitter of 150 ns) occurs due to internal synchronization and processing times. For details, refer to Triggering the Start of Signal Generation Externally (DS4002 Features 🕮).

level Specifies the level which appears at the channel output after the delay has expired respectively a specified trigger occurred. The following symbols are predefined:

Symbol	Description
DS4002_HIGH	high level
DS4002_LOW	low level

instr Specifies the program flow instruction. It must be the same as used in the corresponding ds4002_define_state function call.

The following symbols are predefined:

Symbol	Meaning
DS4002_CONTINUE	Continues with next state.
DS4002_GOTO	Continues with the state, which was defined by using ds4002_define_entry.
DS4002_JUMP	Continues with the state at the given address. This command may not be combined with the DS4002_WAIT directive, and it may not be used within a loop construct, i.e. between a DS4002_LOADCOUNTER and a DS4002_REPEAT instruction!
DS4002_LOADCOUNTER	Continues with next state, save address of next state as a local label, and load the loop counter with the count. This command may not be combined with the DS4002_WAIT directive!
DS4002_REPEAT	Decrements the loop counter; if zero, continue with next state, if not zero, continue with the state following the DS4002_LOADCOUNTER state.

count Specifies the value for the given flow instruction. The following values can be used:

Flow Instruction	Value	Meaning
DS4002_LOADCOUNTER	1 … 256	Specifies the loop counter.
DS4002_JUMP	0 … 255	Specifies the jump address, which is defined by ds4002_define_state
DS4002_CONTINUE DS4002_GOTO DS4002_REPEAT	0	No value required. It must be 0.

Related topics

Examples

Example of Using the Arbitrary Signal Generation Functions.. 71

References

Base Address of the I/O Board.. 15
ds4002_define_state.. 85
Macros.. 15

DS4002_ANGLE

Syntax

```
long DS4002_ANGLE(dsfloat angle)
```

Include file

`ds4002.h`

Purpose

To convert an angle value to time base tics.

Description

You need this macro for the `ds4002_define_state` and `ds4002_update_state` function, if the DS4002 is used in angle-based mode with a base timer cycle from 0 … 360°.

> **Note**
>
> There is no range check for the `angle` parameter.

Parameters

angle Specifies the angle within the range 0 … 179.99°.

Return value

Returns the given angle value as time base value in tics (1 tic = 200 ns).

Example

This example shows how to define a state with an angle-based delay of 90°.

```
ds4002_define_state(
  DS4002_ANGLE(90.0), DS4002_HIGH, 0, DS4002_GOTO, 20)
```

Related topics

References

DS4002_ANGLE2 96
ds4002_define_state 85
ds4002_set_rpm 150
ds4002_update_state 92

DS4002_ANGLE2

Syntax

```
long DS4002_ANGLE2(dsfloat angle)
```

Include file

`ds4002.h`

Purpose

To convert an angle value to time base tics.

Description

You need this macro for the `ds4002_define_state` and `ds4002_update_state` function, if the DS4002 is used in angle-based mode with a base timer cycle from 0 … 720°.

> **Note**
>
> There is no range check for the `angle` parameter.

Parameters

angle Specifies the angle within the range 0 … 359.99°.

Return value

Returns the given angle value as time base value in tics (1 tic = 200 ns).

Example

This example shows how to define a state with an angle-based delay of 90°.

```
ds4002_define_state(
  DS4002_ANGLE2(90.0), DS4002_HIGH, 0, DS4002_GOTO, 20)
```

Related topics

References

DS4002_ANGLE	96
ds4002_define_state	85
ds4002_set_rpm2	152
ds4002_update_state	92

DS4002_DELAY

Syntax

```
long DS4002_DELAY(dsfloat delay)
```

Include file

`ds4002.h`

Purpose

To convert a delay time given in seconds to time base tics.

Description

You need this macro for the `ds4002_define_state` and `ds4002_update_state` functions used in time-based mode.

Parameters

delay Specifies the delay time within the range 0 … 107.374 s. The following symbol is predefined:

Symbol	Meaning
`DS4002_MAXDELAY`	Max. delay time (107.374 s)

Return value

Returns the delay time base value in tics (1 tic = 200 ns).

Example

This example shows how to define a state with a delay time of 1 ms.

```
ds4002_define_state(DS4002_DELAY(0.001), DS4002_HIGH, 0, DS4002_GOTO, 20)
```

Related topics

References

ds4002_define_state.. 85
ds4002_update_state.. 92

DS4002_EXEC_CMD

Syntax

```
void DS4002_EXEC_CMD(
   phs_add_t base,
   long command,
   long channel)
```

Include file

`ds4002.h`

Purpose

To send a command to the DS4002.

Description

This macro executes the specified program data update command for the channel you want to update. The macro waits until the command has been executed. According to channel and command priorities, the execution time can last some seconds, if the controller reaches limitations of the board (refer to Limitations Due to the Controller Processing Time (DS4002 Features 🕮)). In the worst case, there is no response at all, because the controller is overloaded. The controller handles a `DS4002_EXEC_CMD` call with the highest priority. Requests from the channels for new state data will be interrupted. For detailed information on the controller, refer to Basics of the Timing I/O Unit (DS4002 Features 🕮).

> **Note**
>
> The update commands `DS4002_SYNCDATA` and `DS4002_SYNCUSE` have the lowest priority of all update commands and channel requests. Therefore it is possible that the controller do not executes them, if the state machine code uses values near the limitations of the board.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

command Specifies a command code of the DS4002. The following command codes are predefined:

Command Code	Update Mode	Meaning
`DS4002_CMD_COPYWR`		Copies `WRITE` to `COPYWRITE` and `LEN` to `COPYLEN`
`DS4002_CMD_SUBLEN`		Subtracts `COPYLEN` from `LEN`
`DS4002_CMD_SETLEN`		Copies `COPYLEN` to `LEN`
`DS4002_CMD_SETILEN`		Copies `COPYILEN` to `ILEN`
`DS4002_CMD_READTIME`		Copies actual time to `COPYTIME`
`DS4002_CMD_IMMEDIATE`	Immediate mode	Uses new buffer immediately
`DS4002_CMD_NEWDATA`	Next delay mode	Uses new data buffer with next state
`DS4002_CMD_COPYST`		Copies `STATE` to `COPYSTATE`
`DS4002_CMD_WRITETIME`		Uses `COPYTIME` as timer increment
`DS4002_CMD_BLOCKDATA`	Block mode	Uses new buffer after `DS4002_GOTO`
`DS4002_CMD_SYNCDATA`	Synchronous mode	Accepts new data buffer
`DS4002_CMD_SYNCUSE`		Uses new buffers synchronously

Immediate mode The update becomes effective immediately. Even delays which have already started are affected. Long delays may be cut, if the new delay has already expired.

Next delay mode The update becomes effective with the next delay used. Delays which have already started are not affected.

Block mode The update becomes effective after a `DS4002_GOTO` instruction has been executed, which normally occurs after a program cycle has been finished. Delays which have already started are not affected.

Synchronous mode The update becomes effective after a `DS4002_GOTO` instruction has been executed, which normally occurs after a program cycle has been finished. In addition, several channels can be programmed to use the updated state machine code synchronously. Two commands are used here (DS4002_CMD_SYNCDATA, DS4002_CMD_SYNCUSE): The first advances the swinging buffer controller, but the new data does not yet become effective. Only a flag is set indicating that new data is available. This command must be used for all channels to be synchronously updated. Then a second command makes the new data effective for all channels with the above flag, after a DS4002_GOTO instruction has been executed.

> **Note**
>
> The channels have to reach the DS4002_GOTO instruction at the same time in the state machine code to perform a synchronous update.

For further information about the swinging buffer update, refer to Swinging Buffer Principle (DS4002 Features). For information on the update modes, refer to Updating State Parameters (DS4002 Features).

channel Specifies the logical channel number in the range 1 … 8.

Return value

None

Example

This example shows how to use the function in *immediate mode*:

```
…
ds4002_update_state(base, channel, state1, …);
ds4002_update_state(base, channel, state2, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_IMMEDIATE, channel);
```

This example shows how to use the function in *next delay mode*:

```
…
ds4002_update_state(base, channel, state1, …);
ds4002_update_state(base, channel, state2, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_NEWDATA, channel);
```

This example shows how to use the function in *block mode*:

```
…
ds4002_update_state(base, channel, state1, …);
ds4002_update_state(base, channel, state2, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_BLOCKDATA, channel);
```

This example shows how to use the function in *synchronous mode*:

```
…
ds4002_update_state(base, channel1, state1, …);
ds4002_update_state(base, channel1, state2, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_SYNCDATA, channel1);
ds4002_update_state(base, channel2, state3, …);
ds4002_update_state(base, channel2, state4, …);
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_SYNCDATA, channel2);
   /* dummy channel nr  DUMMY_NO */
DS4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_SYNCUSE, DUMMY_NO);
```

Related topics

Basics

Basics of the Timing I/O Unit (DS4002 Features 🕮)
Limitations Due to the Controller Processing Time (DS4002 Features 🕮)

References

Base Address of the I/O Board........ 15
ds4002_define_state........ 85
ds4002_update_state........ 92

DS4002_MASK

Syntax

```
long DS4002_MASK(long channel)
```

Include file

`ds4002.h`

Purpose

To convert a channel number to a bit mask.

Description

Bit masks are needed for all functions which address several channels simultaneously (i.e. `ds4002_start_channels`, `ds4002_define_state`, `ds4002_update_state`).

Parameters

channel	Specifies the channel which should be set in the bit mask.

Return value

This macro returns the bit mask of the specified channel in unsigned long format.

Example

This example shows how to start an output mode application on channels 1 and 3.

```
ds4002_start_channels(
  DS4002_1_BASE, DS4002_MASK(1) + DS4002_MASK(3));
```

Related topics

References

External Triggering

ds4002_ext_trigger_set

Syntax

```
void ds4002_ext_trigger_set (
   phs_addr_t base,
   long trigger,
   long channel,
   long enable)
```

Include file

`ds4002.h`

Purpose

To set the external trigger.

Description

The specified external trigger signal TRIGA or TRIGB is connected to or disconnected from the specified channels.

I/O mapping

For information on the I/O mapping, refer to Triggering the Start of Signal Generation Externally (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

trigger Specifies the external trigger input

Symbol	Description
`DS4002_TRIG_A`	Selects the TRIGA external trigger
`DS4002_TRIG_B`	Selects the TRIGB external trigger

channel Specifies the logical channel number within the range 1 … 8. This channel will be enabled or disabled for external triggering.

enable Enables or disables the external trigger

Symbol	Description
`DS4002_TRIG_DISABLE`	External trigger is disabled
`DS4002_TRIG_ENABLE`	External trigger is enabled

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_ext_trigger_set(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Related topics

References

Base Address of the I/O Board........ 15
ds4002_init........ 19
Macros........ 15

PWM Signal Measurement (PWM2D)

Where to go from here

Information in this section

Example of Using the PWM Signal Measurement Functions

Introduction

The following example demonstrates how to use the PWM signal measurement functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 2 are initialized for PWM analysis. Both channels are read in an interrupt service routine every 1 µs.

Channel 1 gives an average frequency and duty cycle from the last 10 periods.

Channel 2 gives the frequency and duty cycle of the last period, if one has occurred since the last read.

You have to connect the channels 1 … 2 to a frequency generator with variable duty cycle.

```
#include "brtenv.h"
#include "ds4002.h"
```

```
/***********************************************************************
  global variables
************************************************************************/
dsfloat freq1 = 0.0;
dsfloat freq2 = 0.0;
dsfloat duty1 = 0.0;
dsfloat duty2 = 0.0;
long ch1_error = 0;
long ch2_error = 0;
/********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long count;
  ts_timestamp_type ts;
  count = 10;
  ch1_error = ds4002_pwm2d_overl(
    DS4002_1_BASE, 1, count, &len, &freq1, &duty1);
  count = 10;
  ch2_error = ds4002_pwm2d_contig(
   DS4002_1_BASE, 2, count, &len, &freq2, &duty2);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                            /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);              /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_pwm2d_init(DS4002_1_BASE, 1, 0, 0.0);
  ds4002_pwm2d_init(DS4002_1_BASE, 2, 0, 0.0);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_pwm2d_init

Syntax

```
void ds4002_pwm2d_init(
    phs_addr_t base,
    long channel,
    long intlen,
    dsfloat f_min)
```

Include file

`ds4002.h`

Purpose

To initialize the DS4002 channel for PWM signal measurement.

Description

After initialization, the `ds4002_pwm2d_contig` and `ds4002_pwm2d_overl` functions can be used for the specified channel.

I/O mapping

For information on the I/O mapping, refer to PWM Signal Measurement (PWM2D) (DS4002 Features).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

intlen Specifies the number of detected events, at which a host interrupt shall be generated within the range 0 … 511. If no interrupt is requested, the value 0 must be given. The channel(s) which have generated an interrupt can be identified by the `DS4002_INT_STATUS` macro. You can acknowledge the interrupt request by the `DS4002_INT_CLEAR` macro. It is recommended to use `ds4002_pwm2d_contig` within the interrupt service routine, because only this function clears and resets the buffer.

> **Note**
>
> When you have specified 511 as `intlen` parameter, be sure to use the `ds4002_pwm2d_contig` function to clear and reset the buffer in the interrupt service routine. Otherwise each following edge detection will generate another interrupt.

f_min Allows to check for the presence of an input signal. It is used to distinguish between mere slow input signals and the absence of any events. As long as a period of (1/`f_min`) has not yet passed, and no input events have been captured, then `DS4002_EMPTY` is returned by the `ds4002_pwm2d_contig` function. The `ds4002_pwm2d_overl` function returns the old value and `DS4002_NO_ERROR` in this case. After (1/`f_min`) has passed, `DS4002_NO_ERROR` is returned along with a value of 0.0 for `freq`. A duty cycle value of 0.0 is returned, if the input signal remains on low level, a duty cycle value of 1.0 is returned, if the input signal remains on high level.

This feature can be disabled by setting `f_min` to 0.0. In this case, the `ds4002_pwm2d_contig` function returns `DS4002_EMPTY` and the `ds4002_pwm2d_overl` function returns the last measured value at the absence of any events.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_pwm2d_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the PWM Signal Measurement Functions.. 105

References

Base Address of the I/O Board.. 15
ds4002_init.. 19
ds4002_pwm2d_contig.. 108
ds4002_pwm2d_overl... 110
Macros.. 15

ds4002_pwm2d_contig

Syntax

```
int ds4002_pwm2d_contig(
   phs_addr_t base,
   long channel,
   long count,
   long *len,
   dsfloat *freq,
   dsfloat *duty)
```

Include file

`ds4002.h`

Purpose

To measure the average frequency and duty cycle in contiguous mode.

Description

The average frequency and duty cycle of the input signal is computed for the next `count` signal periods, starting at the last unused event, and returned by the `freq` and `duty` parameters. The `*len` parameter returns the number of events that have been actually read. If the buffer contains more than 300 events, the newest data is used for analysis, and the buffer is cleared. If the buffer contains less than the to `count` corresponding number of events, the available events are used.

This function may be used to implement a contiguous PWM analysis. This requires that the function is called at a higher rate than the input events are received. Although, the DS4002's event buffer can temporarily buffer up to 510 events, for example, in case the input rate is not constant.

For information on the contiguous mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 🕮).

The measurement algorithm used is accurate if the PWM period starts with the falling or rising edge of the corresponding PWM signal (asymmetric signal).

The DS4002 can also be used to measure PWM signals that are centered around the middle of the PWM period (symmetric signals). However, the measurement of the PWM frequency of symmetric PWM signals is faulty if the duty cycle of the PWM signal changes during measurement. For details, refer to Limitation for the Measurement of Symmetric PWM Signals (DS4002 Features 🕮).

Note

- One signal period consists of two events.
- The specified channel must have been initialized for PWM analysis by using the `ds4002_pwm2d_init` function.

I/O mapping

For information on the I/O mapping, refer to PWM Signal Measurement (PWM2D) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count Specifies the number of signal periods from which the average frequency and duty cycle are evaluated within the range 1 … 150. This corresponds to the maximum number of 300 events.

len Returns the number of periods that have been actually evaluated.

freq Returns the average frequency measured in Hz.

duty Returns the average duty cycle measured within the range 0.0 … 1.0.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
`DS4002_EMPTY`	The event buffer is empty. For example, no signal is connected to the respective input channel.

Symbol	Description
`DS4002_OVERFLOW`	The event buffer contains more than 300 events. In this case, the newest data is used for analysis, and the buffer is cleared.
`DS4002_INVALID`	Negative frequency values have been measured due to buffer overruns.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
…
long count, len;
dsfloat freq, duty;
ds4002_pwm2d_init ( DS4002_1_BASE, 1, 0, 0.0 );
count = 10;
ds4002_pwm2d_contig( DS4002_1_BASE, 1, count, &len, &freq, &duty );
…
```

The average frequency and duty cycle is computed for the last 10 signal periods of the channel 1 input signal.

Related topics

Examples

Example of Using the PWM Signal Measurement Functions..105

References

Base Address of the I/O Board...15
ds4002_pwm2d_init..106
Macros..15

ds4002_pwm2d_overl

Syntax

```
int ds4002_pwm2d_overl(
   phs_addr_t base,
   long channel,
   long count,
   long *len,
   dsfloat *freq,
   dsfloat *duty)
```

Include file

`ds4002.h`

Purpose

To measure the average frequency and duty cycle in overlapped mode.

Description

The average frequency and duty cycle of the input signal is computed from the last `count` signal periods and returned by the `freq` and `duty` parameters.If the function is called periodically in smaller steps than needed to sample the specified amount of new input data, the intervals being analyzed will overlap. The DS4002's event buffer is used as a circular buffer. Once the buffer has been filled, it always contains the last 512 event data. If the buffer contains less than the to `count` corresponding number of events, the available events are used.

For information on the overlapped mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 🕮).

The measurement algorithm used is accurate if the PWM period starts with the falling or rising edge of the corresponding PWM signal (asymmetric signal).

The DS4002 can also be used to measure PWM signals that are centered around the middle of the PWM period (symmetric signals). However, the measurement of the PWM frequency of symmetric PWM signals is faulty if the duty cycle of the PWM signal changes during measurement. For details, refer to Limitation for the Measurement of Symmetric PWM Signals (DS4002 Features 🕮).

Note

- One signal period consists of two events resulting.
- The specified channel must have been initialized for PWM analysis by using the `ds4002_pwm2d_init` function.

I/O mapping

For information on the I/O mapping, refer to PWM Signal Measurement (PWM2D) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count Specifies the number of signal periods from which the average frequency and duty cycle are evaluated within the range 1 … 250.

len Returns the number of periods that have been actually evaluated.

freq Returns the average frequency measured in Hz.

duty Returns the average duty cycle measured within the range 0.0 … 1.0.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
DS4002_EMPTY	The event buffer is empty. For example, no signal is connected to the respective input channel.
DS4002_INVALID	Negative frequency values have been measured due to buffer overruns.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
…
long count, len;
dsfloat freq, duty;
ds4002_pwm2d_init ( DS4002_1_BASE, 1, 0, 0.0 );
count = 10;
ds4002_pwm2d_overl( DS4002_1_BASE, 1, count, &len, &freq, &duty);
…
```

The average frequency and duty cycle is computed for the last 10 signal periods of the channel 1 input signal.

Related topics

Basics

Limitation for the Measurement of Symmetric PWM Signals (DS4002 Features 📖)
Overlap and Contiguous Read Modes (DS4002 Features 📖)

Examples

Example of Using the PWM Signal Measurement Functions.. 105

References

Base Address of the I/O Board.. 15
ds4002_pwm2d_init.. 106
Macros.. 15

Square-Wave Signal Measurement (F2D)

Where to go from here

Information in this section

Example of Using the Square-Wave Signal Measurement Functions

Introduction

The following example demonstrates how to use the square-wave signal measurement functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 📖).

Description

Channels 1 … 2 are initialized for frequency measurement. Both channels are read in an interrupt service routine every 100 µs.

Channel 1 gives an average frequency from the last 10 periods.

Channel 2 gives the frequency of the last period, if one has occurred since the last read.

You have to connect the channels 1 … 2 to a frequency generator.

```
#include "brtenv.h"
#include "ds4002.h"
```

```
/*************************************************************************
  global variables
*************************************************************************/
dsfloat freq1 = 0.0;
dsfloat freq2 = 0.0;
long ch1_error = 0;
long ch2_error = 0;
/*********************************************************************
  interrupt service routine
**********************************************************************/
void isr_t1()
{
  long count;
  ts_timestamp_type ts;
  count = 10;
  ch1_error = ds4002_f2d_overl(DS4002_1_BASE, 1, count, &len, &freq1);
  count = 1;
  ch2_error = ds4002_f2d_contig(DS4002_1_BASE, 2, count, &len, &freq2);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/*********************************************************************
  main
**********************************************************************/
void main()
{
  init();                            /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);              /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_f2d_init(DS4002_1_BASE, 1, 0, 1.0);
  ds4002_f2d_init(DS4002_1_BASE, 2, 0, 1.0);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_f2d_init

Syntax

```
void ds4002_f2d_init(
   phs_addr_t base,
   long channel,
   long intlen,
   dsfloat f_min)
```

Include file

`ds4002.h`

Purpose

To initialize a channel for frequency measurement.

I/O mapping

For information on the I/O mapping, refer to Square-Wave Signal Measurement (F2D) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

intlen Specifies the number of detected events, at which a host interrupt shall be generated. If no interrupt is requested, the value 0 must be given. The channel(s) which have generated an interrupt can be identified by the `DS4002_INT_STATUS` macro. You can acknowledge the interrupt request by the `DS4002_INT_CLEAR` macro. It is recommended to use `ds4002_f2d_contig` within the interrupt service routine, because this function clears and resets the buffer.

Note

When you have specified 511 as `intlen` parameter, be sure to use the `ds4002_f2d_contig` function to clear and reset the buffer in the interrupt service routine. Otherwise each following edge detection will generate another interrupt.

f_min Allows to check for the presence of an input signal. It is used to distinguish between mere slow input signals and the absence of any events. As long as a period of (1/`f_min`) has not yet passed, and no input events have been captured, then `DS4002_EMPTY` is returned by the `ds4002_f2d_contig` function. The `ds4002_f2d_overl` function returns the old value and `DS4002_NO_ERROR` in this case. After (1/`f_min`) has passed, `DS4002_NO_ERROR` is returned along with a value of 0.0 for `freq`.

This feature can be disabled by setting `f_min` to 0.0. In this case, the `ds4002_f2d_contig` function returns `DS4002_EMPTY` and the `ds4002_f2d_overl` function returns the last measured value at the absence of any events.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_f2d_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the Square-Wave Signal Measurement Functions.. 113

References

Base Address of the I/O Board.. 15
ds4002_f2d_contig.. 116
ds4002_f2d_overl.. 118
ds4002_init... 19
DS4002_INT_CLEAR... 16
DS4002_INT_STATUS.. 17

ds4002_f2d_contig

Syntax

```
int ds4002_f2d_contig(
   phs_addr_t base,
   long channel,
   long count,
   long *len,
   dsfloat *freq)
```

Include file

`ds4002.h`

Purpose

To implement a contiguous frequency measurement.

Description

The average frequency of the input signal is computed for the next `count` signal periods, starting at the last unused event, and returned by the `freq` parameter.

If the buffer contains more than 500 events, the newest data is used for analysis, and the buffer is cleared. If the buffer contains less than `count` events, the available events are used. The `*len` parameter returns the number of events that have been actually read.

This function may be used to implement a contiguous frequency measurement. This requires that the function is called at a higher rate than the input events are received. Although, the DS4002's event buffer can temporarily buffer up to 510 events, for example in case the input rate is not constant.

For information on the contiguous mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 📖).

> **Note**
>
> The specified channel must have been initialized for frequency measurement by using the `ds4002_f2d_init` function.

I/O mapping

For information on the I/O mapping, refer to Square-Wave Signal Measurement (F2D) (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count specifies the number of events to be read within the range 1 … 500.

len Returns the number of events that have been actually read.

freq Returns the average frequency measured in Hz.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
`DS4002_NO_ERROR`	No error occurred
`DS4002_EMPTY`	The event buffer is empty. For example, no signal is connected to the respective input channel.
`DS4002_OVERFLOW`	The event buffer contains more than 510 events. In this case, the newest data is used for analysis, and the buffer is cleared.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
…
int err;
long count, len;
dsfloat freq;
ds4002_f2d_init (DS4002_1_BASE, 1, 0, 0.0);
count = 10;
err = ds4002_f2d_contig (DS4002_1_BASE, 1, count, &len, &freq);
…
```

The average frequency is computed for the last 10 signal periods of the channel 1 input signal.

Related topics

Basics

Overlap and Contiguous Read Modes (DS4002 Features 📖)

Examples

Example of Using the Square-Wave Signal Measurement Functions.. 113

References

Base Address of the I/O Board.. 15
ds4002_f2d_init.. 114
ds4002_init.. 19
Macros.. 15

ds4002_f2d_overl

Syntax

```
int ds4002_f2d_overl(
   phs_addr_t base,
   long channel,
   long count,
   long *len,
   dsfloat *freq)
```

Include file

`ds4002.h`

Purpose

To measure the average frequency in overlapped mode.

Description

The average frequency of the input signal is computed from the last `count` signal periods and returned by the `freq` parameter. If the function is called periodically in smaller steps than needed to sample the specified amount of new input data, the intervals being analyzed will overlap. The DS4002's event buffer is used as a circular buffer. Once the buffer has been filled, it always contains the last 512 event data. If the buffer contains less than `count` events, the available events are used. The `*len` parameter returns the number of events that have been actually read.

For information on the overlapped mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 🕮).

> **Note**
>
> The specified channel must have been initialized for frequency measurement by using the `ds4002_f2d_init` function.

I/O mapping

For information on the I/O mapping, refer to Square-Wave Signal Measurement (F2D) (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count Specifies the number of signal periods from which the average frequency is computed within the range 1 … 511.

len Returns the number of events that have been actually read.

freq Returns the average frequency measured in Hz.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
DS4002_NO_ERROR	No error occurred
DS4002_EMPTY	The event buffer is empty. For example, no signal is connected to the respective input channel.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
…
int err;
long count, len;
dsfloat freq;
ds4002_f2d_init (DS4002_1_BASE, 1, 0, 0.0);
count = 10;
err = ds4002_f2d_overl (DS4002_1_BASE, 1, count, &len, &freq);
…
```

The average frequency is computed for the last 10 signal periods of the channel 1 input signal.

Related topics

Basics

Overlap and Contiguous Read Modes (DS4002 Features 📖)

Examples

Example of Using the Square-Wave Signal Measurement Functions.. 113

References

Base Address of the I/O Board... 15
ds4002_f2d_init.. 114
Macros.. 15

Phase-Shift Measurement

Introduction

The timing I/O unit of the DS4002 provides inputs for the measurement of the average phase shift Δφ for up to 4 signal pairs.

Where to go from here

Information in this section

Information in other sections

Phase-Shift Measurement (DS4002 Features 🕮)
The timing I/O unit of the DS4002 provides inputs for the measurement of the average phase shift.

Example of Using the Phase-Shift Measurement Functions

Introduction

The following example demonstrates how to use the phase-shift measurement functions of the DS4002.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 🕮).

Description

Channels 1 … 2 are initialized for phase measurement. Both channels are read in an interrupt service routine every 100 µs.

The average phase is calculated from the last 10 periods. On both channels the rising edge is used.

You have to connect the channels 1 … 2 to a dual frequency generator with variable phase shift.

```
#include "brtenv.h"
#include "ds4002.h"
/********************************************************************
  global variables
*********************************************************************/
dsfloat phase = 0.0;
/********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long count;
  ts_timestamp_type ts;
  count = 1;
  ds4002_phase_overl(DS4002_1_BASE, 1, 2, count, &len, &phase);
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                           /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);             /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_phase_init(DS4002_1_BASE, 1, DS4002_RISING, 2, DS4002_RISING);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

ds4002_phase_init

Syntax

```
void ds4002_phase_init(
   phs_addr_t base,
   long channel1,
   long edge1,
   long channel2,
   long edge2)
```

Include file

`ds4002.h`

Purpose

To initialize 2 channels for phase-shift measurement.

Description

After initialization `ds4002_phase_overl` function can be used for the specified channels subsequently.The active edges (rising or falling) can be selected by the `edge1` and `edge2` parameters.

The phase-shift measurement can be used for contiguous signals with constant frequency only.

I/O mapping

For information on the I/O mapping, refer to Phase-Shift Measurement (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel1 Specifies the logical number of the first channel within the range 1 … 8. It must be different from `channel2`.

channel2 specifies the logical number of the second channel within the range 1 … 8. It must be different from `channel1`.

edge1 Specifies the active edge of the first channel. The following symbols are predefined:

Symbol	Description
`DS4002_FALLING`	Active on falling edge
`DS4002_RISING`	Active on rising edge

Note

You cannot combine the `DS4002_FALLING` and `DS4002_RISING` symbol.

edge2 Specifies the active edge of the second channel. Use a symbol from the table above.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_phase_init(0x??): Board not initialized!	This error is caused by the `ds4002_read_init` function which is called by `ds4002_phase_init`. The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the Phase-Shift Measurement Functions	121

References

Base Address of the I/O Board	15
ds4002_init	19
ds4002_phase_overl	124
ds4002_read_init	129
Macros	15

ds4002_phase_overl

Syntax

```
int ds4002_phase_overl(
   phs_addr_t base,
   long channel1,
   long channel2,
   long count,
   long *len,
   dsfloat *phase)
```

Include file

`ds4002.h`

Purpose

To compute the average phase-shift between 2 channels in overlapped mode.

Description

The average phase-shift of the `channel2` input signal against the reference signal at `channel1` is computed for `count` signal periods and returned by the `phase` parameter. The active edges (rising or falling) can be selected by the `ds4002_phase_init` function. If the function is called periodically in smaller steps than needed to sample the specified amount of new input data, the intervals being analyzed will overlap. The DS4002's event buffer is used as a circular buffer. Once the buffer has been filled, it always contains the last 512 event data. If the buffer contains less than `count` events, the available event data is used for phase calculation.

For information on the overlapped mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 📖).

Note

The specified channels must have been initialized for phase measurement by using the `ds4002_phase_init` function.

I/O mapping

For information on the I/O mapping, refer to Phase-Shift Measurement (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel1 specifies the logical number of channel 1 within the range 1 … 8.

channel2 Specifies the logical number of channel 2 within the range 1 … 8.

count Specifies the number of signal periods from which the phase-shift is computed within the range 1 … 509.

len Returns the number of events that have been actually read.

phase Returns the average phase-shift measured. The value is scaled in rad and mapped into the interval +π … –π.

Return value

Returns an error code. The following symbol is predefined:

Symbol	Description
`DS4002_NO_ERROR`	No error occurred
`DS4002_EMPTY`	The event buffer is empty. For example, no signal is connected to the respective input channel.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
…
int err;
long count, len;
dsfloat phase;
ds4002_phase_init(DS4002_1_BASE, 1, DS4002_RISING, 2, DS4002_FALLING);
count = 10;
err = ds4002_phase_overl(DS4002_1_BASE, 1, 2, count, &len, &phase);
…
```

The average phase-shift of the falling edge at input channel 2 versus the matching rising edge at input channel 1 is measured for the last 10 periods.

Related topics

Basics

Overlap and Contiguous Read Modes (DS4002 Features 📖)

Examples

Example of Using the Phase-Shift Measurement Functions........ 121

References

Base Address of the I/O Board........ 15
ds4002_phase_init........ 122
Macros........ 15

Event Data Capture

Introduction

To determine the characteristics of arbitrary digital input signals, you can directly access the event buffer of the DS4002. The event buffer holds the direction of captured edges and the time stamps.

Where to go from here

Information in this section

Information in other sections

Event Capture (DS4002 Features 📖)
Using the timing I/O unit and the event buffer, you can also measure and capture arbitrary pulse patterns.

Example of Using the Event Capture Functions

Introduction

The following example demonstrates how to use the event capture functions of the DS4002. You find the relevant files in `<RCP_HIL_InstallationPath>\Demos\Ds100<x>\IOBoards\DS4002\Cust_in`. Use ControlDesk to load and start the application.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 📖).

Description

Channels 1 … 2 are initialized for data acquisition. Both channels are read in an interrupt service routine every 100 µs.

Channel 1 gives an average period from the last 10 events.

Channel 2 gives the last period, if an event has occurred since the last read.

You have to connect the channels 1 … 2 to a frequency generator.

Note

For real-time data capture, use the `custin_4002_hc.cdp` project with ControlDesk.

```
#include "brtenv.h"
#include "ds4002.h"
/************************************************************************
  global variables
************************************************************************/
dsfloat period1 = 0.0;
dsfloat period2 = 0.0;
int ch1_error = 0;
int ch2_error = 0;
/********************************************************************
  interrupt service routine
*********************************************************************/
void isr_t1()
{
  long state[11];
  long time[11];
  long count;
  static long last;
  ts_timestamp_type ts;
  count = 11;
  ch1_error = ds4002_read_overl(DS4002_1_BASE, 1, count, &len, state, time);
  if (count > 1)
    period1 = DS4002_TIME2FLOAT(time[0] - time[count-1]) / (count-1);
  else
    period1 = 0.0;
  count = 1;
  ch2_error = ds4002_read_contiguous(DS4002_1_BASE, 2, &count, state, time);
  if (count == 1)
  {
    period2 = DS4002_TIME2FLOAT(time[0] - last);
    last = time[0];
  }
  else
    period2 = 0.0;
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}
/********************************************************************
  main
*********************************************************************/
void main()
{
  init();                          /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);           /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4002_read_init(DS4002_1_BASE, 1, DS4002_RISING, 0);
  ds4002_read_init(DS4002_1_BASE, 2, DS4002_RISING, 0);
  RTLIB_SRT_START(0.0001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

References

ds4002_init	19
ds4002_read_contiguous	135
ds4002_read_init	129
ds4002_read_overl	137
DS4002_TIME2FLOAT	142

ds4002_read_init

Syntax

```
void ds4002_read_init(
   phs_addr_t base,
   long channel,
   long edge,
   long intlen)
```

Include file

`ds4002.h`

Purpose

To initialize a channel for standard input mode.

Description

The specified channel is initialized for standard input mode, i.e. to use the `ds4002_read_overl`, `ds4002_read_contig` and `ds4002_read_contiguous` functions for event data reading.

I/O mapping

For information on the I/O mapping, refer to Event Capture (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

edge Enables the falling or rising edge detection. The following symbols are predefined:

Symbol	Description
`DS4002_FALLING`	Enables falling edge detection
`DS4002_RISING`	Enables rising edge detection
`DS4002_BOTH`	Enables falling and rising edge detection

intlen Specifies the number of detected events, at which a host interrupt shall be generated within the range 0 … 511. If no interrupt is requested, the value 0 must be given. The channel(s) which have generated an interrupt can be identified by the `DS4002_INT_STATUS` macro. You can acknowledge the interrupt request by the `DS4002_INT_CLEAR` macro. It is recommended to use `ds4002_read_contig` within the interrupt service routine, because this function clears and resets the buffer.

> **Note**
>
> When you have specified 511 as `intlen` parameter, be sure to use the `ds4002_read_contig` function to clear and reset the buffer in the interrupt service routine. Otherwise each following edge detection will generate another interrupt.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_read_init(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Examples

Example of Using the Event Capture Functions.......... 128

References

Base Address of the I/O Board.......... 15
ds4002_init.......... 19
ds4002_read_contig.......... 132
ds4002_read_overl.......... 137
Macros.......... 15

ds4002_read_time

Syntax

```
long ds4002_read_time(
   phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To read the actual DS4002 time.

Description

All DS4002 channels use a common time base, which is generated by a 30-bit counter. For standard time based input and output modes, the counter is incremented by 1 every 200 ns. For the angle-based mode, the counter is incremented by a value representing the rotation speed every 200 ns. The counter wraps around from `0x3fffffff` to `0x0000000`.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

The returned 30-bit value is shifted left 2 bits to represent a valid signed long value in the range from `0x00000000` to `0xfffffffc`. In this way, 2 time values can be subtracted without caring about wraparound or arithmetic overflows. A bit masking of the result is not necessary.

> **Tip**
>
> To convert time values in time base tics to float times or frequencies, use the `DS4002_TIME2FLOAT` or `DS4002_TIME2FREQ` macros.

Execution times

For information, refer to Function Execution Times on page 167.

Example

The following example shows how to calculate the execution time required by `function_x`.

```
Int32 time1, time2;
Float32 dt;
…
time1 = ds4002_read_time(DS4002_1_BASE);
… /* function x() */
time2 = ds4002_read_time(DS4002_1_BASE);
dt = DS4002_TIME2FLOAT(time2 - time1);
```

Related topics

References

Base Address of the I/O Board.. 15
DS4002_TIME2FLOAT.. 142
DS4002_TIME2FREQ.. 143

ds4002_read_contig

Syntax

```
int ds4002_read_contig(
   phs_addr_t base,
   long channel,
   long count,
   long *len,
   long *state,
   long *time)
```

Include file

`ds4002.h`

Purpose

To make event data available for customer specific signal analysis in contiguous mode.

Description

This function is intended to make DS4002 event data available for customer specific signal analysis that cannot be performed by using the standard functions. A maximum number of `count` events are read from the DS4002's event buffer and the corresponding state and time stamp information are returned through the `*state` and `*time` parameter vectors. Event data is stored in increasing order, i.e. time stamps increase with increasing index. The first vector element time[0] contains the time stamp of the first event since the last call to `ds4002_read_contig`. Data input starts at the first event buffer position which has not been read by a previous call to `ds4002_read_contig` and stops either if `count` events have been read, or if the buffer contains no more new events. This allows reading of contiguous segments of event data without overlapping. If the buffer contains less than `count` events, the available events are read. If there were more than `count` events between the last 2 read operations, this function returns the oldest `count` events from the last read operation, beginning with the first event.

For information on the contiguous mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 🕮).

> **Note**
>
> The specified input must have been initialized for input mode by the `ds4002_read_init` function with falling edge detection, rising edge detection, or both enabled.

I/O mapping

For information on the I/O mapping, refer to Event Capture (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count Specifies the number of events to be read within the range 1 … 300.

len Returns the number of events that have been actually read.

state Returns the state information. The memory must be allocated by the calling program with at least `count` words in length.

Value	State
0	Falling edge
1	Rising edge

time Returns the time stamps of the specified events as time base tics. The memory must be allocated by the calling program with at least `count` words in length. To convert the time values in time base tics to float times or frequencies, use the `DS4002_TIME2FLOAT` or `DS4002_TIME2FREQ` macros. To convert the time values to absolute angle, use the `DS4002_TIME2ANGLE` or `DS4002_TIME2ANGLE2` macros.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
`DS4002_NO_ERROR`	No error occurred
`DS4002_EMPTY`	The event buffer is empty. For example, no signal is connected to the respective input channel.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
Int err;
Int32 edge[30];
Int32 time[30];
Float32 period[30];
Int32 j, n, len;
…
err = ds4002_read_init(DS4002_1_BASE, 1, DS4002_RISING, 0);
n = 30;
err = ds4002_read_contig(DS4002_1_BASE, 1, n, &len, edge, time);
j = 0;
for ( i = 0; i < (len-1); i++)
   period[j++] = DS4002_TIME2FLOAT (time[i+1] - time[i]);
…
```

The last 30 events are read from the DS4002's event buffer if available. Then the period duration is computed for each signal period from the rising edge time stamps actually read.

Related topics

Basics

Overlap and Contiguous Read Modes (DS4002 Features 📖)

Examples

Example of Using the Event Capture Functions 128

References

Base Address of the I/O Board 15
ds4002_read_init 129
DS4002_TIME2ANGLE 140
DS4002_TIME2ANGLE2 141
DS4002_TIME2FLOAT 142
DS4002_TIME2FREQ 143
Macros 15

ds4002_read_contiguous

Syntax

```
int ds4002_read_contiguous(
   phs_addr_t base,
   long channel,
   long *count,
   long *state,
   long *time)
```

Include file

`ds4002.h`

Purpose

To make event data available for customer specific signal analysis in contiguous mode (reverse order).

Description

This function is intended to make DS4002 event data available for customer specific signal analysis that cannot be performed by using the standard functions. A maximum number of `*count` events are read from the DS4002's event buffer and the corresponding state and time stamp information are returned through the `*state` and `*time` parameter vectors. Event data is stored in reverse order, i.e. time stamps decrease with increasing index. The first vector element time[0] contains the time stamp of the most recent event. Data input starts at the current event buffer position and stops if `*count` events have been read or the buffer position already read by a previous call to `ds4002_read_contiguous` is reached. This allows reading of contiguous segments of event data without overlapping. If the buffer contains less than `count` events, the available events are read. If there were more than `count` events between the last 2 read operations, this function returns the oldest `count` events from the last read operation, beginning with the latest event.

For further information on the contiguous mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 📖).

> **Note**
>
> The specified input must have been initialized for input mode by the `ds4002_read_init` function with falling edge detection, rising edge detection, or both enabled.

I/O mapping

For information on the I/O mapping, refer to Event Capture (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count Specifies the number of events to be read. After function call it returns the number of actually read events. Valid range is 1 .. 300.

state Returns the state information. The memory must be allocated by the calling program with at least `count` words in length.

Value	State
0	Falling edge
1	Rising edge

time Returns the time stamps of the specified events as time base tics. The memory must be allocated by the calling program with at least `count` words in length. To convert the time values in time base tics to float times or frequencies, use the `DS4002_TIME2FLOAT` or `DS4002_TIME2FREQ` macros. To convert the time values to absolute angle, use the `DS4002_TIME2ANGLE` or `DS4002_TIME2ANGLE2` macros.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
`DS4002_NO_ERROR`	No error occurred
`DS4002_EMPTY`	The event buffer is empty. For example, no signal is connected to the respective input channel.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
Int err;
long edge[30];
long time[30];
dsfloat period[30];
long j, n;
…
err = ds4002_read_init(DS4002_1_BASE, 1, DS4002_RISING, 0);
n = 30;
err = ds4002_read_contiguous(DS4002_1_BASE, 1, &n, edge, time);
j = 0;
for ( i = 0; i < (n-1); i++)
   period[j++] = DS4002_TIME2FLOAT (time[i] - time[i+1]);
…
```

The last 30 events are read from the DS4002's event buffer, if available. Then the period duration is computed for each signal period from the rising edge time stamps actually read.

Related topics

References

Base Address of the I/O Board 15
ds4002_read_init 129
DS4002_TIME2ANGLE 140
DS4002_TIME2ANGLE2 141
DS4002_TIME2FLOAT 142
DS4002_TIME2FREQ 143
Macros 15

ds4002_read_overl

Syntax

```
int ds4002_read_overl(
   phs_addr_t base,
   long channel,
   long count,
   long *len,
   long *state,
   long *time)
```

Include file

`ds4002.h`

Purpose

To make event data available for customer specific signal analysis in overlapped mode.

Description

This function is intended to make event data available for customer specific signal analysis that cannot be performed by using the standard functions. The last `count` events are read from the DS4002's event buffer and the corresponding state and time stamp information are returned through the `*state` and `*time` parameter vectors. Event data is stored in reverse order, i.e. time stamps decrease with increasing index. The first vector element time[0] contains the time stamp of the most recent event. Deviating from the `ds4002_read_contig` function the segments of event data being read may overlap. If the buffer contains less than `count` events, the available events are read. If there were more than `count` events between the last 2 read operations, this function returns always the last `count` events.

For further information on the overlapped mode, refer to Overlap and Contiguous Read Modes (DS4002 Features 🕮).

> **Note**
>
> The specified input must have been initialized for input mode by the `ds4002_read_init` function with falling edge detection, rising edge detection, or both enabled.

I/O mapping

For information on the I/O mapping, refer to Event Capture (DS4002 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

count Specifies the number of events to be read within the range 1 … 511.

len Returns the number of events that have been actually read.

state Returns the state information. The memory must be allocated by the calling program with at least `count` words in length.

Value	State
0	Falling edge
1	Rising edge

time Returns the time stamps of the specified events as time base tics. The memory must be allocated by the calling program with at least `count` words in length. To convert the time values in time base tics to float times or frequencies, use the `DS4002_TIME2FLOAT` or `DS4002_TIME2FREQ` macros. To convert the time values to absolute angle, use the `DS4002_TIME2ANGLE` or `DS4002_TIME2ANGLE2` macros.

Return value

Returns an error code. The following symbols are predefined:

Symbol	Description
DS4002_NO_ERROR	No error occurred
DS4002_EMPTY	The event buffer is empty. For example, no signal is connected to the respective input channel.

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to use the function:

```
Int err;
long edge[22], time[22];
dsfloat freq, duty, prd;
long i, n, len;
…
err = ds4002_read_init(DS4002_1_BASE, 1, DS4002_BOTH, 0);
…
freq = 0.0; duty = 0.0;
n = 22;
err = ds4002_read_overl(DS4002_1_BASE, 1, n, &len, edge, time);
for ( i = 0; i < (len-2); i++)
{
   if (edge[i])  /* true = rising, false = falling edge */
   {
      prd = DS4002_TIME2FLOAT (time[i] - time[i+2]);
      freq += 1 / prd;
      duty += DS4002_TIME2FLOAT (time[i+1] - time[i+2]) / prd;
   }
}
freq = freq / (float) (len-2);
duty = duty / (float) (len-2);
…
```

The average frequency and duty cycle are computed from a segment of 22 events (10 signal periods) of the channel 1 input signal.

Related topics

Basics

Overlap and Contiguous Read Modes (DS4002 Features 🕮)

Examples

Example of Using the Event Capture Functions.. 128

References

Base Address of the I/O Board... 15
ds4002_read_contig.. 132
ds4002_read_init.. 129
DS4002_TIME2ANGLE... 140
DS4002_TIME2ANGLE2... 141
DS4002_TIME2FLOAT... 142
DS4002_TIME2FREQ... 143
Macros.. 15

DS4002_TIME2ANGLE

Syntax

```
dsfloat DS4002_TIME2ANGLE(long time)
```

Include file

`ds4002.h`

Purpose

To convert a timestamp given in long format to an absolute angle given in float.

Description

You need this macro for the `ds4002_read_contig`, `ds4002_read_contiguous` or the `ds4002_read_overl` function, if the DS4002 is used in angle-based mode with a base timer cycle from 0 … 360°, set by `ds4002_set_rpm`.

Parameters

time Specifies the timestamp to be converted.

Return value

This macro returns the time as a float value within the range 0 … 359.99°.

Example

This example shows how to convert the timestamp from the last event on channel 1.

```
ds4002_read_overl(
 DS4002_1_BASE, 1, &count, len, &state, &time);
angle = DS4002_TIME2ANGLE(time);
```

Related topics

References

ds4002_read_contig.. 132
ds4002_read_contiguous.................................... 135
ds4002_read_overl... 137
ds4002_set_rpm.. 150
DS4002_TIME2ANGLE2.. 141

DS4002_TIME2ANGLE2

Syntax

```
dsfloat DS4002_TIME2ANGLE2(long time)
```

Include file

`ds4002.h`

Purpose

To convert a timestamp given in long format to an absolute angle given in float.

Description

You need this macro for the `ds4002_read_contig`, `ds4002_read_contiguous` or the `ds4002_read_overl` function, if the DS4002 is used in angle-based mode with a base timer cycle from 0 … 720°, set by `ds4002_set_rpm2`.

Parameters

time Specifies the timestamp to be converted.

Return value

This macro returns the time as a float value within the range 0 … 719.99°.

Example

This example shows how to convert the timestamp from the last event on channel 1.

```
ds4002_read_overl(
  DS4002_1_BASE, 1, &count, len, &state, &time);
angle = DS4002_TIME2ANGLE2(time);
```

Related topics

References

ds4002_read_contig..132
ds4002_read_contiguous..135
ds4002_read_overl..137
ds4002_set_rpm2..152
DS4002_TIME2ANGLE..140

DS4002_TIME2FLOAT

Syntax

```
dsfloat DS4002_TIME2FLOAT(long time)
```

Include file

`ds4002.h`

Purpose

To convert a time stamp difference given in long format to time given in seconds.

Description

With this function, you can calculate the time difference, which have been read before by using `ds4002_read_contig`, `ds4002_read_contiguous` or `ds4002_read_overl`. It can be used in time-based mode.

Parameters

time Specifies the timestamp differences for the calculation.

Return value

This function returns the time in seconds.

Example

This example shows how to calculate the time difference of the last two edges.

```
ds4002_read_overl(DS4002_1_BASE, 1, &count, len, state, time);
time_delta = DS4002_TIME2FLOAT(time[0] - time[1]);
```

Related topics

References

ds4002_read_contig..132
ds4002_read_contiguous..135
ds4002_read_overl..137
DS4002_TIME2FREQ..143

DS4002_TIME2FREQ

Syntax

```
dsfloat DS4002_TIME2FREQ(long time)
```

Include file

`ds4002.h`

Purpose

To convert a timestamp difference given in long format to a frequency given in 1/s.

Description

With this function, you can calculate the frequency of timestamp differences, which have been read before by using `ds4002_read_contig`, `ds4002_read_contiguous` or `ds4002_read_overl`. It can be used in time-based mode.

Parameters

time Specifies the timestamp differences for the calculation.

Return value

This function returns the timestamp difference in float format as 1/s.

Example

This example shows how to calculate the frequency of the last two edges.

```
ds4002_read_overl(DS4002_1_BASE, 1, &count, len, state, time);
freq = DS4002_TIME2FREQ(time[0] - time[1]);
```

Related topics

References

Angle-Based Mode

Where to go from here

Information in this section

Example of Using Angle-Based Functions

Introduction

The following example demonstrates how to use functions of the DS4002 in angle-based mode.

Tip

If you want to use a C-coded program in your RTI model, you have to implement the program as an S-function. For detailed information, refer to Inserting Custom C/C++ Code (RTI and RTI-MP Implementation Guide 📖).

Description

In this example the time base is interpreted as a 0 … 720° angle value, representing two complete rotations of an engine crankshaft.

Channel 1 generates a crankshaft position signal with a 58/2 pattern, which means that during one rotation 60 pulses are generated, but the pulses number 59 and 60 are suppressed.

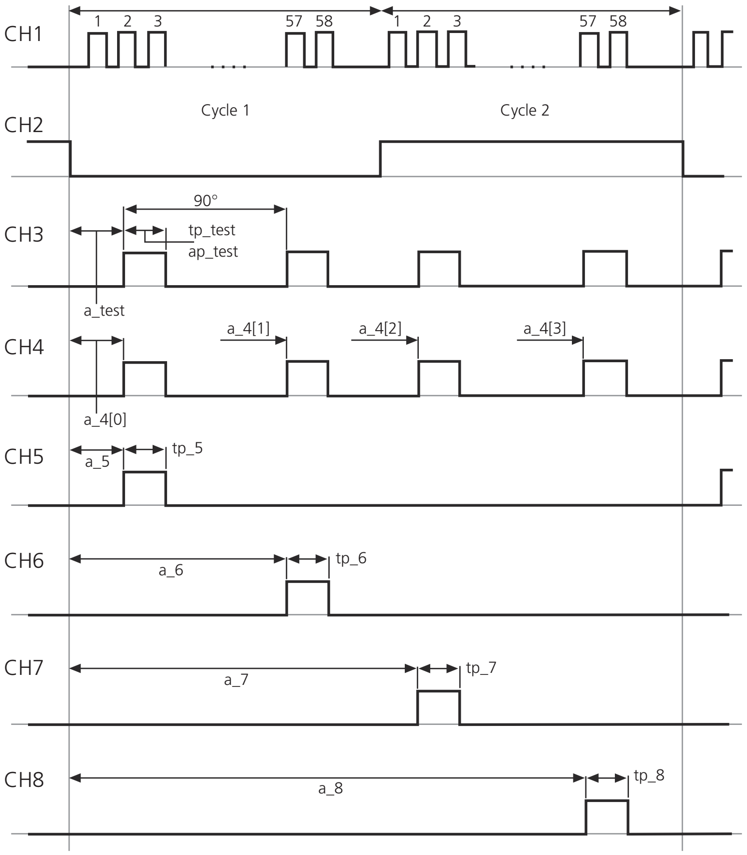
Channel 2 acts as a main timer and generates an additional position signal, indicating either the first rotation cycle (low) or the second (high). At angle 0° a host interrupt is generated.

Channel 3 generates a test signal which can be connected to channels 4 … 8 to simulate injection or ignition pulses.

Channel 4 captures an ignition pulse pattern consisting of up to 8 single ignition pulses. The angles of the rising edges of the pulses are returned (active high pulses are assumed).

Channels 5 … 8 each capture an injection pulse. The angle of the rising edge of the pulse and the pulse width (in seconds) are returned (active high pulses are assumed).

All channels are updated in an interrupt service routine every 1.0 ms.



You have to connect channel 3 to channels 4 … 8.

When using the test signal of channel 3, channels 5 … 8 measure each of the pulses as specified in the variable `n_test` instead of only 1 pulse.

```
#include "brtenv.h"        /* basic real time environment */
#include "ds4002.h"     /* DS4002 constants and macros */
/*******************************************************************************
  global variables
*******************************************************************************/
dsfloat rpm      = 50000;                                  /* initial values */
dsfloat tp_test  = 20e-6;
dsfloat ap_test;
dsfloat a_test   = 10.0;
int     n_test   = 8;
```

```
dsfloat a_4[8]   = {0,0,0,0,0,0,0,0};
dsfloat tp_5     = 0;
dsfloat a_5      = 0;
dsfloat tp_6     = 0;
dsfloat a_6      = 0;
dsfloat tp_7     = 0;
dsfloat a_7      = 0;
dsfloat tp_8     = 0;
dsfloat a_8      = 0;
dsfloat scale;             /* scaling value for converting timestamps to time */
long  state[16];                                 /* data array for input data */
long  time[16];                                  /* data array for input data */
long  i, count, len;
int   addr[4];                      /* list of update addresses for channel 3 */
/*******************************************************************************
  interrupt service routine
*******************************************************************************/
void isr_t1()                             /* timer1 interrupt service routine */
{
  ts_timestamp_type ts;
  /* update channel 3 (test signal generator) */
  ds4002_update_state(DS4002_1_BASE, 3, addr[0],
     DS4002_ANGLE2(a_test),                             /* after angle a_test */
     DS4002_HIGH,                                          /* set output high */
     DS4002_LOADCOUNTER,                      /* continue with next state and */
     n_test-1);                                          /* load loop counter */
  ap_test = 6.0 * tp_test * rpm;
    /* conversion: angle = 360 deg * t/period with period = 60/rpm */
  ds4002_update_state(DS4002_1_BASE, 3, addr[1],
     DS4002_ANGLE2(ap_test),                           /* after angle ap_test */
     DS4002_LOW,                                            /* set output low */
     DS4002_CONTINUE,                             /* continue with next state */
     0);                                     /* no loop counter or jump value */
  ds4002_update_state(DS4002_1_BASE, 3, addr[2],
     DS4002_ANGLE2(720/n_test - ap_test),          /* n_test pulses per cycle */
     DS4002_HIGH,                                          /* set output high */
     DS4002_REPEAT,         /* decrement loop counter. If not zero, goto local
                               entry label. Else, continue with next state    */
     0);                                     /* no loop counter or jump value */
  ds4002_update_state(DS4002_1_BASE, 3, addr[3],
     DS4002_ANGLE2(ap_test),                                 /* after ap_test */
     DS4002_LOW,                                            /* set output low */
     DS4002_GOTO,                         /* goto entry point (= first state) */
     0);                                     /* no loop counter or jump value */
  /* set time base frequency, store resulting new scale factor */
  scale = ds4002_set_rpm2(DS4002_1_BASE, rpm );
  /* advance swinging buffer for use with next delay */
  ds4002_EXEC_CMD(DS4002_1_BASE, DS4002_CMD_NEWDATA, 3);
  /* calculate angle and pulse width from input channel 5 */
  count = 3;
  ds4002_read_overl(DS4002_1_BASE, 5, count, &len, state, time);
  if (count == 3)
  {
    if (state[0] == 0) i=0; else i=1;
    /* time[i] now points to last falling edge*/
    a_5 = DS4002_TIME2ANGLE2(time[i+1]);
    tp_5 = (time[i] - time[i+1]) * scale;
  }
  /* calculate angle and pulse width from input channel 6 */
  count = 3;
  ds4002_read_overl(DS4002_1_BASE, 6, count, &len, state, time);
  if (count == 3)
  {
    if (state[0] == 0) i=0; else i=1;
    /* time[i] now points to last falling edge*/
    a_6 = DS4002_TIME2ANGLE2(time[i+1]);
    tp_6 = (time[i] - time[i+1]) * scale;
  }
  /* calculate angle and pulse width from input channel 7 */
  count = 3;
  ds4002_read_overl(DS4002_1_BASE, 7, count, &len, state, time);
  if (count == 3)
  {
    if (state[0] == 0) i=0; else i=1;
    /* time[i] now points to last falling edge*/
    a_7 = DS4002_TIME2ANGLE2(time[i+1]);
    tp_7 = (time[i] - time[i+1]) * scale;
  }
  /* calculate angle and pulse width from input channel 8 */
  count = 3;
  ds4002_read_overl(DS4002_1_BASE, 8, count, &len, state, time);
  if (count == 3)
  {
    if (state[0] == 0) i=0; else i=1;
    /* time[i] now points to last falling edge*/
    a_8 = DS4002_TIME2ANGLE2(time[i+1]);
    tp_8 = (time[i] - time[i+1]) * scale;
  }
  ts_timestamp_read(&ts);
  host_service(1, &ts);
}

void channel2_intserv()
{
  /* add your own code for cycle driven activities here */
  /* calculate angles of up to 8 ignition pulses from channel 4 */
  count = 9;
  ds4002_read_overl(DS4002_1_BASE, 4, &count, state, time);
  if (count == 9)
  {
    a_4[0] = DS4002_TIME2ANGLE2(time[7]);
    a_4[1] = DS4002_TIME2ANGLE2(time[6]);
    a_4[2] = DS4002_TIME2ANGLE2(time[5]);
    a_4[3] = DS4002_TIME2ANGLE2(time[4]);
    a_4[4] = DS4002_TIME2ANGLE2(time[3]);
    a_4[5] = DS4002_TIME2ANGLE2(time[2]);
    a_4[6] = DS4002_TIME2ANGLE2(time[1]);
    a_4[7] = DS4002_TIME2ANGLE2(time[0]);
  }
}
/**********************************************************************
  main
**********************************************************************/
void main()
{
  init();                           /* basic hardware initialization */
  ds4002_init(DS4002_1_BASE);            /* initialize DS4002 board */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
```

```
install_phs_int_vector(       /* initialize interrupt controllers */
   DS4002_1_BASE,   /* board base address                         */
   2,               /* slave ICU input
                       0 = ILEN interrupt in input mode
                           (check INT register for channel numbers)
                       1 = channel 1 in output mode
                       2 = channel 2 in output mode               */
   channel2_intserv );              /* address of service routine */
/* ch1: crankshaft pulses */
ds4002_output_init();                /* prepare program variables */
ds4002_define_entry();             /* entry point = program start */
ds4002_define_state(
   DS4002_WAIT,                /* wait for trigger from channel 2 */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
            /* do not combine DS4002_WAIT and DS4002_LOADCOUNTER! */
ds4002_define_state(
   DS4002_ANGLE2(3.0),                             /* after 3 deg */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_LOADCOUNTER,            /* continue with next state and */
   57);                                      /* load loop counter */
ds4002_define_state(
   DS4002_ANGLE2(3.0),                             /* after 3 deg */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_ANGLE2(3.0),                             /* after 3 deg */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_REPEAT,/* decrement loop counter. If not zero, goto local
                      entry label. Else, continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_ANGLE2(3.0),                             /* after 3 deg */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_GOTO,               /* goto entry point (= first state) */
   0);                           /* no loop counter or jump value */
ds4002_load_states(DS4002_1_BASE, 1);
                                /* download program for channel 1 */
/* ch2: main timer */
ds4002_output_init();                /* prepare program variables */
ds4002_define_state(
   0,                                      /* as soon as possible */
   DS4002_LOW,                                  /* set output low */
   DS4002_MASK(1) + DS4002_MASK(3) + DS4002_INTERRUPT,
             /* trigger channels 1 and 3, generate host interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_entry();              /* entry point = second state */
ds4002_define_state(
   DS4002_ANGLE2(180),  /* after 180 deg (must be less than 360!) */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
```

```
ds4002_define_state(
   (0x40000000 - DS4002_ANGLE2(180)) & 0x1fffffff,
           /* complete 360 degree cycle to avoid rounding effects */
   DS4002_HIGH,                                /* set output high */
   DS4002_MASK(1),        /* trigger channel 1, no host interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   DS4002_ANGLE2(180),  /* after 180 deg (must be less than 360!) */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
ds4002_define_state(
   (0x40000000 - DS4002_ANGLE2(180)) & 0x1fffffff,
           /* complete 360 degree cycle to avoid rounding effects */
   DS4002_LOW,                                  /* set output low */
   DS4002_MASK(1) + DS4002_MASK(3) + DS4002_INTERRUPT,
             /* trigger channels 1 and 3, generate host interrupt */
   DS4002_GOTO,              /* goto entry point (= second state) */
   0);                           /* no loop counter or jump value */
ds4002_load_states(DS4002_1_BASE, 2);
                                /* download program for channel 2 */
/* ch3: test output */
ds4002_output_init();                /* prepare program variables */
ds4002_define_entry();             /* entry point = program start */
ds4002_define_state(
   DS4002_WAIT,                /* wait for trigger from channel 2 */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
            /* do not combine DS4002_WAIT and DS4002_LOADCOUNTER! */
addr[0] = ds4002_define_state(
   DS4002_ANGLE2(a_test),                   /* after angle a_test */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_LOADCOUNTER,            /* continue with next state and */
   n_test-1);                                /* load loop counter */
ap_test = 6.0 * tp_test * rpm;
  /* conversion: angle = 360 deg * t/period with period = 60/rpm */
addr[1] = ds4002_define_state(
   DS4002_ANGLE2(ap_test),                       /* after ap_test */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_CONTINUE,                   /* continue with next state */
   0);                           /* no loop counter or jump value */
addr[2] = ds4002_define_state(
   DS4002_ANGLE2(720/n_test - ap_test),/* n_test pulses per cycle */
   DS4002_HIGH,                                /* set output high */
   0,                              /* do not trigger or interrupt */
   DS4002_REPEAT,/* decrement loop counter. If not zero, goto local
                      entry label. Else, continue with next state */
   0);                           /* no loop counter or jump value */
addr[3] = ds4002_define_state(
   DS4002_ANGLE2(ap_test),                       /* after ap_test */
   DS4002_LOW,                                  /* set output low */
   0,                              /* do not trigger or interrupt */
   DS4002_GOTO,               /* goto entry point (= first state) */
   0);                           /* no loop counter or jump value */
  ds4002_load_states(DS4002_1_BASE, 3);
                                  /* download program for channel 3 */
  /* init channels 4 to 8 for input mode */
  ds4002_read_init(DS4002_1_BASE, 4, DS4002_RISING, 0);
  ds4002_read_init(DS4002_1_BASE, 5, DS4002_BOTH, 0);
  ds4002_read_init(DS4002_1_BASE, 6, DS4002_BOTH, 0);
  ds4002_read_init(DS4002_1_BASE, 7, DS4002_BOTH, 0);
  ds4002_read_init(DS4002_1_BASE, 8, DS4002_BOTH, 0);
  ds4002_set_rpm2(DS4002_1_BASE , 0 );           /* freeze time base */
  ds4002_set_rpm2(DS4002_1_BASE , -1  );          /* reset time base */
  ds4002_start_channels(DS4002_1_BASE,      /* start channels 1 to 3 */
     DS4002_MASK(1) + DS4002_MASK(2) + DS4002_MASK(3));
  scale = ds4002_set_rpm2(DS4002_1_BASE , rpm );  /* start time base */
  RTLIB_SRT_START(0.001, isr_t1); /* initialize sampling clock timer */
  RTLIB_INT_ENABLE();
  for (;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

References

DS4002_ANGLE2	96
ds4002_define_entry	89
ds4002_define_state	85
ds4002_output_init	84
ds4002_read_init	129
ds4002_read_overl	137
ds4002_set_rpm2	152
DS4002_TIME2ANGLE2	141
ds4002_update_state	92

ds4002_set_rpm

Syntax

```
dsfloat ds4002_set_rpm(
   phs_addr_t base,
   dsfloat rpm)
```

Include file

`ds4002.h`

Purpose

To set the time base to angle-based mode with an angle width of 360°.

Description

When using the angle-based mode, this function can be used to modify the speed of the time base. The `rpm` parameter is scaled and written to the time base accumulator, so that one full cycle (`0x00000000` to `0x3fffffff`) is performed within 1/rpm minutes, thus representing an angle from 0 … 360°.

To reset the time base accumulator via the `rpm` parameter, you can use the following values:

Value of rpm	Meaning
0.3	Resets the time base accumulator to normal mode (increment = 1).
< 0	Resets the time base accumulator without changing the actual accumulator increment.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

rpm Specifies the speed of the time base within the range 0 … 292968.4, including zero. The resolution is about 0.28 rpm.

Return value

This function returns a float value which can be used to convert time stamps to absolute time (in seconds). The time stamps can be read by using the `ds4002_read_contig` function.

Example

This example shows how to use this function.

```
dsfloat scale, rpm = 50000;
…
scale = ds4002_set_rpm ( DS4002_1_BASE , rpm );/* start time base */
…
```

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

References

Base Address of the I/O Board........ 15
ds4002_init........ 19
ds4002_read_contig........ 132
ds4002_set_rpm2........ 152
Macros........ 15

ds4002_set_rpm2

Syntax

```
dsfloat ds4002_set_rpm2(
   phs_addr_t base,
   dsfloat rpm)
```

Include file

`ds4002.h`

Purpose

To set the time base to angle-based mode with an angle width of 720°.

Description

When using the angle-based mode, this function can be used to modify the speed of the time base. The `rpm` parameter is scaled and written to the time base accumulator, so that one full cycle (`0x00000000` to `0x3fffffff`) is performed within 2/rpm minutes, thus representing an angle from 0 … 720°.

To reset the time base accumulator via the `rpm` parameter, you can use the following values:

Value of rpm	Meaning
0.6	Resets the time base accumulator to normal mode (increment = 1).
< 0	Resets the time base accumulator without changing the actual accumulator increment.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

rpm Specifies the speed of the time base within the range 0 … 585936.8, including zero. The resolution is about 0.56 rpm.

Return value

This function returns a float value which can be used to convert time stamps to absolute time (in seconds). The time stamps can be read by using the `ds4002_read_contig` function.

Example

This example shows how to use this function.

```
dsfloat scale, rpm = 50000;
…
scale = ds4002_set_rpm2 ( DS4002_1_BASE , rpm );/* start time base */
…
```

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

References

Base Address of the I/O Board 15
ds4002_apu_velocity_write 157
ds4002_init 19
ds4002_read_contig 132
ds4002_read_contiguous 135
ds4002_set_rpm 150
Macros 15

Time Base Distribution

Introduction

You can use the time-base connector to distribute the time base of one DS4002 to other I/O boards.

Note

To use the RTLib functions for controlling the time-base connector, board revision DS4002-04 and higher are required.

Where to go from here

Information in this section

Information in other sections

Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04) (DS4002 Features 🕮)
You can implement the angle-based mode on a single DS4002 or an angle-based mode that is synchronized with other I/O boards (only for DS4002 with a board revision as of DS4002-04).

ds4002_apu_master_detect

Syntax

```
int ds4002_apu_master_detect(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To detect a DS4002 or DS5001, which is connected to the time-base connector and initialized as master.

> **Note**
>
> - This function can be used only for board revision DS4002-04 and higher.
> - This function must not be used in conjunction with a DS2210, since this board does not support the detection of the master.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

Returns the status of the master detection. The following symbols are predefined:

Symbol	Meaning
`DS4002_MASTER_FOUND`	There is a DS4002 specified as master.
`DS4002_NO_MASTER_FOUND`	There is no DS4002 specified as master.

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds4002_apu_master_detect(??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the **`ds4002_init`** function.
-206	Error	ds4002_apu_master_detect(??): DS4002 board revision 4 or higher required!	The current DS4002 board has a revision number less than 4. The functions of the time-base connector can be used only for board revision DS4002-04 and higher.

Related topics

References

Base Address of the I/O Board...15
ds4002_apu_mode_set...156
ds4002_init...19

ds4002_apu_mode_set

Syntax

```
void ds4002_apu_mode_set(
   phs_addr_t base,
   long mode)
```

Include file

`ds4002.h`

Purpose

To specify the DS4002 as time-base bus master or slave.

Description

In the master mode the DS4002 will calculate the engine position and supplies the result to the time-base connector, from which slaves (a DS4002, DS5001 or DS2210 in slave mode) can read it. The internal time base of the DS4002 is selected and the increment register is cleared. The timebase stops.

In the slave mode the engine position is read from the time-base connector. The external time base is selected and the increment register is cleared.

Note

- This function can be used only for board revision DS4002-04 and higher.
- Do not configure a DS4002 as the time-base master if the network also contains one or more DS2210 boards. Otherwise, the DS2210 board(s) will not work correctly.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

mode Specifies the mode. The following symbols are predefined:

Symbol	Meaning
`DS4002_SLAVE`	Slave mode
`DS4002_MASTER`	Master mode

Return value None

Messages The following messages are defined:

ID	Type	Message	Description
-50	Error	ds4002_apu_mode_set(??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.
-206	Error	ds4002_apu_mode_set(??): DS4002 board revision 4 or higher required!	The current DS4002 board has a revision number less than 4. The functions of the time-base connector can be used only for board revision DS4002-04 and higher.

Related topics References

Base Address of the I/O Board........ 15
ds4002_apu_master_detect........ 155
ds4002_apu_start........ 158

ds4002_apu_velocity_write

Syntax

```
void ds4002_apu_velocity_write(
   phs_addr_t base,
   dsfloat vel)
```

Include file `ds4002.h`

Purpose To update the angle velocity.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

vel Specifies the angle velocity within the range 0 … 61,359 rad/s.

Return value None

Messages

The following message is defined:

ID	Type	Message	Description
-207	Error	ds4002_apu_velocity_write(??): board is not in APU master mode!	The DS4002 has not been specified as APU master. Use `ds4002_apu_mode_set` to specify the DS4002 as master.

Related topics

References

ds4002_apu_start

Syntax

```
void ds4002_apu_start(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To start the time base distribution via the time-base bus.

Description

This functions starts the engine position phase accumulation of the time-base connector.

Note

- Before you can call this function, you must set the DS4002 to master mode using `ds4002_apu_mode_set`.
- The engine position phase accumulation needs an initial value for the angle velocity. You can specify it using `ds4002_apu_velocity_write`.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-205	Error	ds4002_apu_start(??): No APU velocity value set!	The crankshaft angle velocity has not been specified. Use `ds4002_apu_velocity_write` to specify the velocity.
-207	Error	ds4002_apu_start(??): board is not in APU master mode!	The DS4002 has not been specified as master. Use `ds4002_apu_mode_set` to specify the DS4002 as master.

Related topics

References

Base Address of the I/O Board........ 15
ds4002_apu_mode_set........ 156
ds4002_apu_stop........ 160
ds4002_apu_velocity_write........ 157

ds4002_apu_position_clear

Syntax

```
void ds4002_apu_position_clear(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To clear the engine position.

Description

The engine position will be set to 0.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

None

Related topics

References

Base Address of the I/O Board........ 15
ds4002_apu_position_read........ 160

ds4002_apu_position_read

Syntax

```
void ds4002_apu_position_read(
   phs_addr_t base,
   dsfloat *pos)
```

Include file

`ds4002.h`

Purpose

To read the current engine position.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

pos Returns the address of the current engine position value. It is measured in rad within the range 0 … 4π.

Return value

None

Related topics

References

Base Address of the I/O Board.. 15
ds4002_apu_position_clear.. 159

ds4002_apu_stop

Syntax

```
void ds4002_apu_stop(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To stop the time-base distribution.

Description

This function stops the engine phase accumulation of the time-base connector.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

None

Related topics

References

Base Address of the I/O Board.. 15
ds4002_apu_start.. 158

Input Signal Filtering

Introduction

You can filter the timing I/O channels and the external trigger inputs.

Where to go from here

Information in this section

ds4002_disable_filter

Syntax

```
void ds4002_disable_filter(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To disable the input filter for the timing I/O channels and the external trigger inputs.

Description

All input signals are fed directly to the channel capture units. Events which occur faster than the board controller can handle are lost. If rising and falling edge detection are enabled, then fast changing signals might produce successive time stamps with rising edges, for example.

For further information, refer to Input Signal Filtering (DS4002 Features 🕮).

Note

The input filter affects all channels and external trigger inputs.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_disable_filter(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

Input Signal Filtering (DS4002 Features 🕮)

References

ds4002_enable_filter

Syntax

```
void ds4002_enable_filter(phs_addr_t base)
```

Include file

`ds4002.h`

Purpose

To enable the input filter for the timing I/O channels and the external trigger inputs.

Description

The reaction on an external trigger signal is delayed for 3.2 µs. This means for a 50% duty cycle square-wave signal that there is a bandwidth limitation at about 156 kHz.

For further information, refer to Input Signal Filtering (DS4002 Features 🕮).

Note

The input filter affects all channels and external trigger inputs.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds4002_enable_filter(0x??): Board not initialized!	The DS4002 has not been initialized by a preceding call to the `ds4002_init` function.

Execution times

For information, refer to Function Execution Times on page 167.

Related topics

Basics

Input Signal Filtering (DS4002 Features 📖)

References

Base Address of the I/O Board.. 15
ds4002_init.. 19
Macros.. 15

Bit I/O

ds4002_bit_in

Syntax

```
void ds4002_bit_in(
   phs_addr_t base,
   long channel,
   long *state)
```

Include file

`ds4002.h`

Purpose

To read the state of a channel.

I/O mapping

For information on the I/O mapping, refer to Timing I/O Unit (DS4002 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 15.

channel Specifies the logical channel number in the range 1 … 8.

state returns the state of the specified channel:

Value	Description
1	Current state is high
0	Current state is low

Return value

None

Execution times

For information, refer to Function Execution Times on page 167.

Example

This example shows how to read the state of channel 1:

```
…
ds4002_bit_in (DS4002_1_BASE, 1, &state);
if (state == 1)
…
```

Related topics

References

Base Address of the I/O Board........ 15
Macros........ 15

Function Execution Times

Introduction

To give you the mean function execution times and basic information on the test environment used.

Where to go from here

Information in this section

Information on the Test Environment

Introduction

The execution times of the C functions can vary, since they depend on different factors. The measured execution times are influenced by the test environment used.

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<xxxxx> 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:

	DS1006
CPU clock	2.6 GHz / 3.0 GHz
Bus clock	133 MHz

Measured Execution Times

Introduction

Execution times are available for the following RTLib units:

- Initialization on page 168
- Time measurement on page 168
- Digital I/O unit on page 168
- Timing I/O unit on page 169

Initialization

The following execution time has been measured for the initialization function:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_init`	69.23 µs	65.01 µs

Time measurement

The following execution time has been measured for the time measurement function:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_read_time`	1.85 µs	1.99 µs

Digital I/O unit

The following execution times have been measured for the functions of the digital I/O unit:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_dio_init`	0.02 µs	0.01 µs
`ds4002_dio_initialize`	0.02 µs	0.02 µs
`ds4002_in32`	0.58 µs	0.58 µs
`ds4002_out32`	0.02 µs	0.01 µs

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
ds4002_dio_bit_in	0.58 µs	0.58 µs
ds4002_dio_bit_out	0.59 µs	0.59 µs

Timing I/O unit

The following execution times have been measured for the functions of the timing I/O unit:

- Signal generation:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
1-Phase PWM Signal Generation		
ds4002_pwm_init	6.49 µs	6.44 µs
ds4002_pwm_int_init	6.49 µs	6.43 µs
ds4002_pwm_update	1.65 µs	1.85 µs
3-Phase PWM Signal Generation		
ds4002_pwm3_init	19.78 µs	19.72 µs
ds4002_pwm3_int_init	21.87 µs	21.81 µs
ds4002_pwm3_update	7.28 µs	7.58 µs
ds4002_pwm3_int_update	7.82 µs	8.14 µs
Square-Wave Signal Generation		
ds4002_d2f_init	6.48 µs	6.43 µs
ds4002_d2f_int_init	6.48 µs	6.43 µs
ds4002_d2f_update	1.95 µs	1.94 µs
Monoflop Signal Generation		
ds4002_mono_init	7.23 µs	7.18 µs
ds4002_delayed_mono_int_init	7.24 µs	7.19 µs
ds4002_mono_start	1.18 µs	1.18 µs
ds4002_mono_update	1.74 µs	1.74 µs
ds4002_delayed_mono_int_update	1.93 µs	1.93 µs

- Signal measurement:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
PWM Signal Measurement		
ds4002_pwm2d_init	6.42 µs	6.39 µs
ds4002_pwm2d_contig	4.973 + c[1] · 1.141 µs	5.786 + c[1] · 1.140 µs
ds4002_pwm2d_overl	5.727 + c[1] · 1.140 µs	5.808 + c[1] · 1.140 µs
Square-Wave Signal Measurement		
ds4002_f2d_init	6.42 µs	6.40 µs
ds4002_f2d_contig	4.35 µs	4.82 µs

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_f2d_overl`	5.26 µs	5.23 µs
Phase-Shift Measurement		
`ds4002_phase_init`	12.95 µs	12.95 µs
`ds4002_phase_overl`	6.093 + c[1] · 1.563 µs	6.182 + c[1] · 1.563 µs
Event Capture		
`ds4002_read_init`	6.40 µs	6.38 µs
`ds4002_read_contig`	2.149 + c[1] · 0.572 µs	2.140 + c[1] · 0.572 µs
`ds4002_read_overl`	2.354 + c[1] · 0.572 µs	2.340 + c[1] · 0.572 µs

[1] c is the number of data values to be read.

- Angle-based functions:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_set_rpm`	1.68 µs	1.18 µs
`ds4002_set_rpm2`	1.68 µs	1.19 µs

- Input signal filtering:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_enable_filter`	0.60 µs	0.60 µs
`ds4002_disable_filter`	0.60 µs	0.60 µs

- Bit I/O:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds4002_bit_in`	0.59 µs	0.58 µs