DS5001 Digital Waveform Capture 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

Contents

##

##

About This Reference

Contents

This RTLib Reference (Real-Time Library) gives detailed descriptions of the C functions needed to program a DS5001 Digital Waveform Capture 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.

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.

Macros

Introduction

The base address of an I/O board in a PHS-bus-based system has to be defined by using the `DSxxxx_n_BASE` macro.

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

Board Initialization

Introduction

Before you can use the DS5001 board, you have to perform the initialization process.

ds5001_init

Syntax

```
void ds5001_init(phs_addr_t base)
```

Include file

`ds5001.h`

Purpose

To initialize the DS5001.

Description

All DS5001 registers are initialized to default values.

Capture and edge detection of all channels are disabled.

Note

- This function must be called before any of the DS5001 functions can be used.
- The initialization function of the processor board must be called before the DS5001 board's initialization function.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
201	Error	ds5001_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.
-184	Error	ds5001_init(0x??): Board not found!	No DS5001 could be found at the specified PHS-bus address. Check if the `DSxxxx_n_BASE` macro corresponds to the I/O board used.
-185	Error	ds5001_init(0x??): Memory allocation error!	The allocation of some dynamic memory for internal data storage has failed.
-53	Warning	ds5001_init(0x??): Jumper setup is not matching SW default initialization! STP register: 0x???????? instead of 0x????????.	The value of the STP register could not be verified because the DS5001 jumper setting is not correct.

Execution times

For information, refer to Function Execution Times on page 71.

Timing I/O Unit

Where to go from here

Information in this section

PWM Signal Measurement (PWM2D)

Introduction

You can use the following functions to analyze a pulse width modulated (PWM) signal. In a PWM analysis the average frequency and duty cycle of an input signal are computed.

Where to go from here

Information in this section

Information in other sections

PWM Signal Measurement (PWM2D) (DS5001 Features 📖)
The DS5001 timing I/O unit allows the measurement of average frequency and the duty cycle of pulse-width modulated (PWM) signals.

ds5001_pwm2d_init

Syntax

```
int ds5001_pwm2d_init(
   phs_addr_t base,
   int channel,
   dsfloat level,
   int intlen,
   dsfloat f_min)
```

Include file

`ds5001.h`

Purpose

To initialize a DS5001 channel for PWM analysis.

Description

The specified DS5001 channel is initialized for PWM analysis. The `ds5001_pwm2d_contig` and `ds5001_pwm2d_overl` functions can be used for this channel.

I/O mapping

For details on the I/O mapping, refer to PWM Signal Measurement (PWM2D) (DS5001 Features 🕮).

Parameters

base Specifies the PHS-bus base address, refer to DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number in the range 1 … 16.

> **Note**
>
> If you use an older DS5001 board (board revision less than DS5001-06), channel 16 is not available if zero frequency detection is enabled, because it is used to read the current time.

level Specifies the trigger level within the range –10.0 V … +10.0 V.

intlen Contains the number of detected events within the range 0 … 511, at which a host interrupt shall be generated. If no interrupt is requested, the value 0 must be given. When using 511, be sure that you read the event buffer immediately after the first interrupt by using the `ds5001_pwm2d_contig` function. While the event buffer contains 511 events, 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 `DS5001_EMPTY` is returned by the `ds5001_pwm2d_contig` function. The `ds5001_pwm2d_overl` function returns the old value and `DS5001_NO_ERROR` in this case. After (`1/f_min`) has passed, `DS5001_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 `ds5001_pwm2d_contig` function returns `DS5001_EMPTY` and the `ds5001_pwm2d_overl` function returns the last measured value at the absence of any events and `DS5001_NO_ERROR`.

Return value

The following value is returned:

Return Value	Meaning
`DS5001_NO_ERROR`	No error occurred during initialization

This return value is only kept for compatibility purposes. In case of an error this function will perform an exit.

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds5001_pwm2d_init(0x??): Board not initialized!	The DS5001 has not been initialized by a preceding call to the `ds5001_init` function.

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_pwm2d_contig

Syntax

```
int ds5001_pwm2d_contig(
   phs_addr_t base,
   int channel,
   long count,
   long *len,
   dsfloat *freq,
   dsfloat *duty_cycle)
```

Include file

`ds5001.h`

Purpose

To compute the average frequency and duty cycle of an input signal.

Description

The average frequency and duty cycle of the input signal are 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 510 events (buffer overflow), 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 DS5001'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 read mode, refer to Event Buffer Read Modes (DS5001 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 DS5001 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 (DS5001 Features 📖).

Note

- One signal period consists of two events.
- The specified channel must have been initialized for PWM analysis by using the `ds5001_pwm2d_init` function.

I/O mapping

For details on the I/O mapping, refer to PWM Signal Measurement (PWM2D) (DS5001 Features 📖).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

channel Specifies the channel number in the range 1 … 16.

Note

If you use an older DS5001 board (board revision less than DS5001-06), channel 16 is not available if zero frequency detection is enabled, because it is used to read the current time.

count Specifies the number of signal periods from which the average frequency and duty cycle are evaluated within the range 1 … 255.

len Returns the number of periods that have been actually used for computation.

freq Returns the average frequency computed measured in Hz.

duty_cycle Returns the duty cycle computed within the range 0 … 1.

Return value

This function returns the following values:

Return Value	Meaning
`DS5001_NO_ERROR`	No error while measuring.
`DS5001_EMPTY`	The event buffer is empty (< 4 events). For example, there is no signal connected to the respective input channel.
`DS5001_FIFO_OVERFLOW`	There is a FIFO overflow.
`DS5001_EVENT_OVERFLOW`	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.
`DS5001_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 71.

Example

This example shows how to use the function:

```
int Ierr;
Int32 len;
Float32 freq, duty;
void isr_t1()   /* timer1 interrupt service routine */
{
   Ierr = ds5001_pwm2d_contig(DS5001_1_BASE, 1, 10, &len, &freq, &duty);
}
main()
{
   init();
   ds5001_init(DS5001_1_BASE);
   ds5001_pwm2d_init(DS5001_1_BASE, 1, 1.4, 0, 0.0);
   /* initialize sampling clock timer */
   RTLIB_SRT_START(DT, isr_t1);
   while(1)
   {
      RTLIB_BACKGROUND_SERVICE();
   }
}
```

The average frequency and duty cycle of the channel 1 input are computed for the next 10 signal periods in the buffer which have not been evaluated yet.

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 🕮)

References

Base Address of the I/O Board	7
ds5001_init	9
ds5001_pwm2d_init	12

ds5001_pwm2d_overl

Syntax

```
int ds5001_pwm2d_overl(
   phs_addr_t base,
   int channel,
   long count,
   long *len,
   dsfloat *freq,
   dsfloat *duty_cycle)
```

Include file

`ds5001.h`

Purpose

To compute the average frequency and duty cycle of an input signal.

Description

The average frequency and duty cycle of the input signal are computed from the last `count` signal periods and returned by the `*freq` and `*duty` parameters. A signal period starts with a rising edge.

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 DS5001 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 (DS5001 Features 🕮).

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 DS5001'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.

For information on the overlapped read mode, refer to Event Buffer Read Modes (DS5001 Features 🕮).

Note

- The specified channel must have been initialized for PWM analysis by using the `ds5001_pwm2d_init` function.
- The internal event buffer counter is not decremented. Therefore, do not use this function in an Intlen interrupt service routine.

I/O mapping

For details on the I/O mapping, refer to PWM Signal Measurement (PWM2D) (DS5001 Features 🕮).

Parameters

base Specifies the PHS-bus base address, refer to DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number in the range 1 … 16.

Note

If you use an older DS5001 board (board revision less than DS5001-06), channel 16 is not available if zero frequency detection is enabled, because it is used to read the current time.

count Specifies the number of signal periods from which the average frequency and duty cycle are evaluated within the range 1 … 255.

len Returns the number of periods that have been actually used for computation.

freq Returns the average frequency computed in Hz.

duty_cycle Returns the duty cycle computed within the range 0 … 1.

Return value

This function returns the following values:

Return Value	Meaning
`DS5001_NO_ERROR`	No error while measuring.
`DS5001_EMPTY`	The event buffer is empty (< 4 events). For example, there is no signal connected to the respective input channel.
`DS5001_FIFO_OVERFLOW`	There is a FIFO overflow.
`DS5001_EVENT_OVERFLOW`	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
…
int Ierr;
Int32 len;
Float32 freq, duty;
void isr_t1()   /* timer1 interrupt service routine */
{
   Ierr = ds5001_pwm2d_overl(DS5001_1_BASE, 1, 10, &len, &freq, &duty);
…
}
main()
{
   init();
   ds5001_init(DS5001_1_BASE);
   ds5001_pwm2d_init(DS5001_1_BASE, 1, 1.4, 0, 0.0);
   /* initialize sampling clock timer */
   RTLIB_SRT_START(DT, isr_t1);
   while(1)
   {
      RTLIB_BACKGROUND_SERVICE();
   }
}
```

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

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 📖)

References

ds5001_init.. 9
ds5001_pwm2d_init... 12

Square-Wave Signal Measurement (F2D)

Introduction

You can use the following functions to compute the average frequency of a signal.

> **Note**
>
> You have to initialize the DS5001 with the `ds5001_init` function before you can use one of these functions.

Where to go from here

Information in this section

Information in other sections

Square-Wave Signal Measurement (F2D) (DS5001 Features 📖)
The DS5001 timing I/O unit allows the measurement of the average frequency of square-wave signals on up to 16 channels.

ds5001_f2d_init

Syntax

```
int ds5001_f2d_init(
   phs_addr_t base,
   int channel,
   dsfloat level,
   int intlen,
   dsfloat f_min)
```

Include file

`ds5001.h`

Purpose

To initialize a channel for frequency measurement.

Description

The specified DS5001 channel is initialized for frequency measurement. The `ds5001_f2d_contig` and `ds5001_f2d_overl` functions can be used for this channel.

I/O mapping

For details on the I/O mapping, refer to Square-Wave Signal Measurement (F2D) (DS5001 Features 🕮).

Parameters

base Specifies the PHS-bus base address, refer to DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number in the range 1 … 16.

> **Note**
>
> If you use an older DS5001 board (board revision less than DS5001-06), channel 16 is not available if zero frequency detection is enabled, because it is used to read the current time.

level Specifies the trigger level within the range –10.0 V … +10.0 V.

intlen Contains the number of detected events within the range 0 … 511, at which a host interrupt shall be generated. If no interrupt is requested, the value 0 must be given. When using 511, be sure that you read the event buffer immediately after the first interrupt by using the `ds5001_f2d_contig` function. While the event buffer contains 511 events, 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 `DS5001_EMPTY` is returned by the `ds5001_f2d_contig` function. The `ds5001_f2d_overl` function returns the old value and `DS5001_NO_ERROR` in this case. After (`1/f_min`) has passed, `DS5001_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 `ds5001_f2d_contig` function returns `DS5001_EMPTY` and the `ds5001_f2d_overl` function returns the last measured value at the absence of any events and `DS5001_NO_ERROR`.

Return value

The following value is returned:

Return Value	Meaning
`DS5001_NO_ERROR`	No error occurred during initialization

This return value is only kept for compatibility purposes. In case of an error this function will perform an exit.

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds5001_f2d_init(0x??): Board not initialized!	The DS5001 has not been initialized by a preceding call to the `ds5001_init` function.

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_f2d_contig....22
ds5001_f2d_overl....25
ds5001_init....9

ds5001_f2d_contig

Syntax

```
int ds5001_f2d_contig(
   phs_addr_t base,
   int channel,
   long count,
   long *len,
   dsfloat *freq)
```

Include file

`ds5001.h`

Purpose

To compute the average frequency of an input signal.

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 DS5001's event buffer is empty (for example, no signal connected to the respective input channel), the function returns the error code `DS5001_EMPTY`. If the buffer contains more than 510 events (buffer overflow), `DS5001_OVERFLOW` is returned. In this case 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 evaluated.

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 DS5001'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 read mode, refer to Event Buffer Read Modes (DS5001 Features 🕮).

Note

The specified channel must have been initialized for frequency measurement by using the `ds5001_f2d_init` function.

I/O mapping

For details on the I/O mapping, refer to Square-Wave Signal Measurement (F2D) (DS5001 Features 🕮).

Parameters

base Specifies the PHS-bus base address, refer to DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number in the range 1 … 16.

Note

If you use an older DS5001 board (board revision less than DS5001-06), channel 16 is not available if zero frequency detection is enabled, because it is used to read the current time.

count Specifies the number of signal periods from which the average frequency is computed.

len Returns the number of periods that have been actually used for computation.

freq Returns the average frequency in Hz.

Return value

This function returns the following values:

Return Value	Meaning
`DS5001_NO_ERROR`	No error while measuring.
`DS5001_EMPTY`	The event buffer is empty (< 2 events). For example, there is no signal connected to the respective input channel.
`DS5001_FIFO_OVERFLOW`	There is a FIFO overflow.
`DS5001_EVENT_OVERFLOW`	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.

Return Value	Meaning
DS5001_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 71.

Example

This example shows how to use the function:

```
…
int Ierr;
Int32 len;
Float32 freq;
void isr_t1()   /* timer1 interrupt service routine */
{
   Ierr = ds5001_f2d_contig(DS5001_1_BASE, 1, 10, &len, &freq);
…
}
main()
{
   init();
   ds5001_init(DS5001_1_BASE);
   ds5001_f2d_init(DS5001_1_BASE, 1, 1.4, 0, 0.0);
   /* initialize sampling clock timer */
   RTLIB_SRT_START(DT, isr_t1);
   while(1)
   {
      RTLIB_BACKGROUND_SERVICE();
   }
}
```

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

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 🕮)

References

ds5001_f2d_init...20
ds5001_init..9

ds5001_f2d_overl

Syntax

```
int ds5001_f2d_overl(
   phs_addr_t base,
   int channel,
   long count,
   long *len,
   dsfloat *freq)
```

Include file

`ds5001.h`

Purpose

To compute the average frequency of an input signal.

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 DS5001'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.

For information on the overlapped read mode, refer to Event Buffer Read Modes (DS5001 Features 🕮).

Note

- The specified channel must have been initialized for frequency measurement by using the `ds5001_f2d_init` function.
- The internal event buffer counter is not decremented. Therefore, do not use this function in an Intlen interrupt service routine.

I/O mapping

For details on the I/O mapping, refer to Square-Wave Signal Measurement (F2D) (DS5001 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

channel Specifies the channel number in the range 1 … 16.

> **Note**
>
> If you use an older DS5001 board (board revision less than DS5001-06), channel 16 is not available if zero frequency detection is enabled, because it is used to read the current time.

count Specifies the number of signal periods from which the average frequency is computed.

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

freq Returns the average frequency in Hz.

Return value

This function returns the following values:

Return Value	Meaning
DS5001_NO_ERROR	No error while measuring.
DS5001_EMPTY	The event buffer is empty (< 2 events). For example, there is no signal connected to the respective input channel.
DS5001_FIFO_OVERFLOW	There is a FIFO overflow.
DS5001_EVENT_OVERFLOW	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
…
int Ierr;
Int32 len;
Float32 freq;
void isr_t1()   /* timer1 interrupt service routine */
{
   Ierr = ds5001_f2d_overl(DS5001_1_BASE, 1, 10, &len, &freq);
…
}
main()
{
   init();
   ds5001_init(DS5001_1_BASE);
   ds5001_f2d_init(DS5001_1_BASE, 1, 1.4, 0, 0.0);
   /* initialize sampling clock timer */
   RTLIB_SRT_START(DT, isr_t1);
   while(1)
   {
      RTLIB_BACKGROUND_SERVICE();
   }
}
```

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

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 📖)

References

Base Address of the I/O Board........ 7
ds5001_f2d_init........ 20
ds5001_init........ 9

Phase-Shift Measurement

Introduction

You can use the following functions to compute the average phase-shift of 2 input signals.

Where to go from here

Information in this section

Information in other sections

ds5001_phase_init

Syntax

```
void ds5001_phase_init(
   phs_addr_t base,
   int channel1,
   dsfloat level1,
   int channel2,
   dsfloat level2,
   int edge)
```

Include file

`ds5001.h`

Purpose

To initialize 2 DS5001 channels for phase measurement.

Description

The specified DS5001 channels are initialized for phase measurement. The active edges (falling or rising) can be selected by the `edge` parameter. Only identical edges can be selected for both channels. The `ds5001_phase_overl` function can be used for these channels subsequently.

Note

You must specify 2 different channels for `channel1` and `channel2`.

I/O mapping

For details on the I/O mapping, refer to Phase-Shift Measurement (DS5001 Features 🕮).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel1 Specifies the 1st channel number within the range 1 … 16.

channel2 Specifies the 2nd channel number within the range 1 … 16.

level1 Specifies the trigger level of 1st channel within the range -10.0 V … +10.0 V.

level2 Specifies the trigger level of 2nd channel within the range -10.0 V … +10.0 V.

edge Specifies the active edges. The following symbols are predefined:

Predefined Symbol	Description
`DS5001_FALLING`	Falling edge active
`DS5001_RISING`	Rising edge active

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Example

For an example, refer to `ds5001_phase_overl` on page 30.

Related topics

References

ds5001_phase_overl...30

ds5001_phase_overl

Syntax

```
int ds5001_phase_overl(
   phs_addr_t base,
   int channel1,
   int channel2,
   long count,
   long *len,
   dsfloat *phase)
```

Include file

`ds5001.h`

Purpose

To compute the average phase shift between two input signals.

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 `ds5001_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 DS5001'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 read mode, refer to Event Buffer Read Modes (DS5001 Features 🕮).

> **Note**
>
> The specified channels must have been initialized for phase-shift measurement by using the `ds5001_phase_init` function.

I/O mapping

For details on the I/O mapping, refer to Phase-Shift Measurement (DS5001 Features 🕮).

Parameters

base Specifies the PHS bus base address, see `DSxxxx_n_BASE Macros` (refer to DSxxxx_n_BASE Macros on page 7).

channel1 Specifies the 1st channel number.

channel2 Specifies the 2nd channel number.

count Specifies the number of signal periods within the range 1 … 509.

len Returns the number of events that have been actually used for computation.

phase Returns the phase shift computed. It is scaled in rad and mapped into the interval –π … +π.

Return value

This function returns the following values:

Return Value	Meaning
DS5001_NO_ERROR	No error while measuring.
DS5001_EMPTY	The event buffer is empty (< 3 events). For example, there is no signal connected to the respective input channel.
DS5001_FIFO_OVERFLOW	There is a FIFO overflow.
DS5001_EVENT_OVERFLOW	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
…
int Ierr;
Int32 len;
Float32 phase;
void isr_t1()   /* timer1 interrupt service routine */
{
   Ierr = ds5001_phase_overl(DS5001_1_BASE, 1, 2, 10, &len, &phase);
…
}
main()
{
   init();
   ds5001_init(DS5001_1_BASE);
   ds5001_phase_init(DS5001_1_BASE, 1, 1.4, 2, 1.4, DS5001_RISING);
   /* initialize sampling clock timer */
   RTLIB_SRT_START(DT, isr_t1);
   while(1)
   {
      RTLIB_BACKGROUND_SERVICE();
   }
}
```

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

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 📖)

References

ds5001_init.. 9
ds5001_phase_init.. 28

Incremental Encoder Measurement

Introduction

You can use the following functions to simulate an incremental encoder counter.

Where to go from here

Information in this section

Information in other sections

Incremental Encoder Measurement (DS5001 Features 📖)
The DS5001 timing I/O unit is able to capture digital phase signals from incremental encoder sensors on up to 8 pairs of consecutive channels.

ds5001_enc_init

Syntax

```
int ds5001_enc_init(
   phs_addr_t base,
   int channel,
   dsfloat level1,
   dsfloat level2)
```

Include file

`ds5001.h`

Purpose

To initialize 2 successive channels to be used as an incremental sensor encoder.

Description

The specified DS5001 channel and the subsequent channel are initialized to be used as an incremental sensor encoder. The `ds5001_enc` and `ds5001_enc_clr` functions can be used for these channels subsequently. The incremental encoder counter for the specified channel (implemented as a global variable) is cleared.

Note

Only odd channel numbers can be specified (for example, 1, 3, … 15), because the incremental encoder functions uses pairs of subsequent input channels.

I/O mapping

For details on the I/O mapping, refer to Incremental Encoder Measurement (DS5001 Features 📖).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number within the range 1 … 15 but only odd numbers, for example, 1, 3, …

level1 Specifies the trigger level for the 1st channel within the range -10.0 V … +10.0 V.

level2 Specifies the trigger level for the 2nd channel within the range -10.0 V … +10.0 V.

Return value

The following value is returned:

Return Value	Meaning
`DS5001_NO_ERROR`	No error occurred during initialization

This return value is only kept for compatibility purposes. In case of an error this function will perform an exit.

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds5001_enc_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 71.

Related topics

References

ds5001_enc 35
ds5001_enc_clr 36

ds5001_enc

Syntax

```
int ds5001_enc(
   phs_addr_t base,
   int channel,
   long *count)
```

Include file

`ds5001.h`

Purpose

To process the events captured.

Description

All events captured since the previous call to `ds5001_enc` are processed. The incremental encoder counter for the specified channel is updated to the current value, which is returned by the `*count` parameter.

> **Note**
>
> Only odd channel numbers can be specified (for example, 1, 3, … 15), because the incremental encoder functions uses pairs of subsequent input channels.

This function requires an execution time of 1.0 µs per event. It has been measured with the event buffer being completely filled, i.e. each invocation of `ds5001_enc` had to read out and evaluate the entire event buffer (512 events). Since rising as well as falling edges are evaluated, a theoretical maximum input frequency of approximately 500 kHz can be achieved.

> **Note**
>
> The specified channel must have been initialized for incremental encoder operation by using the `ds5001_enc_init` function.

I/O mapping

For details on the I/O mapping, refer to Incremental Encoder Measurement (DS5001 Features 🕮).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number within the range 1 … 15 but only odd numbers, for example, 1, 3, …

count Returns the current value of the incremental encoder counter.

Return value

This function returns the following values:

Return Value	Meaning
DS5001_NO_ERROR	No error while measuring.
DS5001_FIFO_OVERFLOW	There is a FIFO overflow.
DS5001_EVENT_OVERFLOW	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.
DS5001_OVERFLOW	The event buffer contains more than 511 events. In this case, the buffer is cleared and the `count` value remains unchanged.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
…
ds5001_enc_init(DS5001_1_BASE, 1, 1.4, 1.4);
for (;;)
{
   if ((error = ds5001_enc(DS5001_1_BASE, 1, &count)) != DS5001_NO_ERROR)
   exit();
}
…
```

The counter value of an incremental sensor connected to input channels 1 and 2 is periodically updated as long as the `ds5001_enc` function is executed successfully.

Related topics

References

ds5001_enc_init..33

ds5001_enc_clr

Syntax

```
void ds5001_enc_clr(
   phs_addr_t base,
   int channel)
```

Include file

`ds5001.h`

Purpose

To clear an incremental encoder counter.

Description

The incremental encoder counter corresponding to the specified channel is cleared.

> **Note**
>
> Only odd channel numbers can be specified (for example, 1, 3, … 15), because the incremental encoder functions uses pairs of subsequent input channels.

The specified channel must have been initialized for incremental encoder operation by using the `ds5001_enc_init` function.

I/O mapping

For details on the I/O mapping, refer to Incremental Encoder Measurement (DS5001 Features 📖).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number within the range 1 … 15 but only odd numbers, for example, 1, 3, …

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_enc_init..33

Event Data Capture

Introduction

You can use the following functions to read event data for a customer specific signal analysis.

Where to go from here

Information in this section

Information in other sections

Event Data Capture (DS5001 Features 📖)
The timing I/O unit is able to capture aperiodic (arbitrary) signals on up to 16 channels and convert them into events.

ds5001_read_init

Syntax

```
void ds5001_read_init(
   phs_addr_t base,
   int channel,
   int edge,
   dsfloat level,
   int intlen)
```

Include file

`ds5001.h`

Purpose

To initialize a channel for event data capture input mode.

Description

The specified DS5001 channel is initialized for event data capture input mode, for example, to use the `ds5001_read_overl` and `ds5001_read_contig` functions to read event data.

I/O mapping

For details on the I/O mapping, refer to Event Data Capture (DS5001 Features 🕮).

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

channel Specifies the channel number within the range 1 … 16.

edge Enables the edge detection. The following symbols are predefined:

Predefined Symbol	Description
`DS5001_FALLING`	Falling edges will be detected
`DS5001_RISING`	Rising edges will be detected
`DS5001_BOTH`	Falling and rising edges will be detected

level Specifies the trigger level within the range –10.0 V … +10.0 V.

intlen Contains the number of detected events within the range 0 … 511, at which a host interrupt shall be generated. If no interrupt is requested, the value 0 must be given. When using 511, be sure that you read the event buffer immediately after the first interrupt by using the `ds5001_read_contig` function. While the event buffer contains 511 events, each following edge detection will generate another interrupt.

Return value

None

Messages

The following message is defined:

ID	Type	Message	Description
-50	Error	ds5001_read_init(0x??): Board not initialized!	The DS5001 has not been initialized by a preceding call to the `ds5001_init` function.

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

Base Address of the I/O Board 7
ds5001_init 9
ds5001_read_contig 40
ds5001_read_overl 43

ds5001_read_contig

Syntax

```
int ds5001_read_contig(
   phs_addr_t base,
   int channel,
   long count,
   long *len,
   long *state,
   long *time)
```

Include file

`ds5001.h`

Purpose

To read events from the DS5001's event buffer.

Description

This function is intended to make DS5001 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 DS5001's event buffer and the corresponding state and time stamp information are returned through the `*state` and `*time` vectors. The internal event counter is decremented. 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 `ds5001_read_contig`. Data input starts at the first event buffer position which has not been read by a previous call to `ds5001_read_contig` and stops either if `count` events have been read, or if the buffer contains no more new events. If the buffer contains less than `count` events, the available events are read.

This function may be used to implement a contiguous reading of segments of event data without overlapping. This requires that the function is called at a higher rate than the input events are received. Although, the DS5001'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 read mode, refer to Event Buffer Read Modes (DS5001 Features 🕮).

Note

- The specified channel must have been initialized by using the `ds5001_read_init` function for input mode with falling edge detection, rising edge detection, or both enabled.
- The `*state` and `*time` vectors must be allocated by the calling program with at least `count` words in length.

I/O mapping

For details on the I/O mapping, refer to Event Data Capture (DS5001 Features 🕮).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number within the range 1 … 16.

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 Specifies the pointer to an array of returned 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 Specifies the pointer to an array of returned 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 `DS5001_TIME2FLOAT` or `DS5001_TIME2FREQ` macros. To convert the time values to absolut angle, use the `DS5001_TIME2ANGLE` or `DS5001_TIME2ANGLE2` macros.

Return value

This function returns the following values:

Return Value	Meaning
DS5001_NO_ERROR	No error while measuring.
DS5001_EMPTY	The event buffer is empty (< 4 events). For example, there is no signal connected to the respective input channel.
DS5001_FIFO_OVERFLOW	There is a FIFO overflow.
DS5001_EVENT_OVERFLOW	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
int err;
long edge[30], time[30];
dsfloat period[30];
long j, n, len;
…
err = ds5001_read_init(DS5001_1_BASE, 1, DS5001_RISING, 1.4, 0);
n = 30;
…
err = ds5001_read_contig(DS5001_1_BASE, 1, n, &len, edge, time);
j = 0;
for (i = 0; i < (len-1); i++)
   period[j++] = DS5001_TIME2FLOAT(time[i+1] - time[i]);
…
```

The last 30 events are read from the DS5001's event buffer, if available. Then the period duration is computed for each signal period from the rising edge time stamps actually read. Use the `ds5001_read_contig` function within an interrupt service routine.

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 🕮)

References

ds5001_read_init.. 39
DS5001_TIME2FLOAT.. 48

ds5001_read_overl

Syntax

```
int ds5001_read_overl(
   phs_addr_t base,
   int channel,
   long count,
   long *len,
   long *state,
   long *time)
```

Include file

`ds5001.h`

Purpose

To read events from the DS5001's event buffer.

Description

This function is intended to make DS5001 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 DS5001's event buffer and the corresponding state and time stamp information are returned through the `*state` and `*time` 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 `ds5001_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.

For information on the overlapped read mode, refer to Event Buffer Read Modes (DS5001 Features 🕮).

Note

- The specified channel must have been initialized by using the `ds5001_read_init` function for input mode with falling edge detection, rising edge detection, or both enabled.
- The `*state` and `*time` vectors must be allocated by the calling program with at least `count` words in length.
- The internal event buffer counter is not decremented. Therefore, do not use this function in an Intlen service routine.

I/O mapping

For details on the I/O mapping, refer to Event Data Capture (DS5001 Features 🕮).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number within the range 1 … 16.

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

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

state Specifies the pointer to an array of returned 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 Specifies the pointer to an array of returned 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 `DS5001_TIME2FLOAT` or `DS5001_TIME2FREQ` macros. To convert the time values to absolut angle, use the `DS5001_TIME2ANGLE` or `DS5001_TIME2ANGLE2` macros.

Return value

This function returns the following values:

Return Value	Meaning
`DS5001_NO_ERROR`	No error while measuring.
`DS5001_EMPTY`	The event buffer is empty (< 4 events). For example, there is no signal connected to the respective input channel.
`DS5001_FIFO_OVERFLOW`	There is a FIFO overflow.
`DS5001_EVENT_OVERFLOW`	There is a read and write access onto the same event buffer location (read-pointer = write-pointer). In this case, the read data is invalid.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
int err;
long edge[22];
long time[22];
dsfloat freq, duty, prd;
long i, n, len;
…
err = ds5001_read_init(DS5001_1_BASE, 1, DS5001_BOTH, 1.4, 0);
…
freq = 0.0;
duty = 0.0;
n = 22;
…
err = ds5001_read_overl(DS5001_1_BASE, 1, n, &len, edge, time);
for (i = 0; i < (len-2); i++)
{
   if (edge[i])  /* true = rising, false = falling edge */
   {
      prd =DS5001_TIME2FLOAT (time[i] – time[i+2]);
      freq += 1 / prd;
      duty += DS5001_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. Use the `ds5001_read_overl` function within an interrupt service routine.

Related topics

Basics

Event Buffer Read Modes (DS5001 Features 📖)

References

ds5001_read_contig.. 40
ds5001_read_init.. 39
DS5001_TIME2FLOAT.. 48

ds5001_timebase_read

Syntax

```
long ds5001_timebase_read(phs_addr_t base)
```

Include file

`ds5001.h`

Purpose

To read the current value of the timebase counter.

Description

All DS5001 channels use a common time base, which is generated by a 31-bit counter. For standard time based input and output modes, the counter is incremented by 1 every 25 ns. For the angle-based mode, the counter is incremented by a value representing the rotation speed every 25 ns. The counter wraps around from `0x7FFFFFFF` to `0x0000000`.

> **Note**
>
> If you use this function on a DS5001 board older than revision 6, channel 16 is not available for other features.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

Return value

This functions returns a right-aligned 31-bit time-base value.

> **Tip**
>
> To convert time values in time base tics to float times or frequencies, use the `DS5001_TIME2FLOAT` or `DS5001_TIME2FREQ` macros.

Execution times

For information, refer to Function Execution Times on page 71.

Example

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

```
Int32 time1, time2;
Float32 dt;
…
time1 = ds5001_timebase_read(DS5001_1_BASE);
… /* function x() */
time2 = ds5001_timebase_read(DS5001_1_BASE);
dt = DS5001_TIME2FLOAT(time2 - time1);
```

Related topics

References

Base Address of the I/O Board.. 7
DS5001_TIME2FLOAT.. 48
DS5001_TIME2FREQ.. 49

DS5001_TIME2ANGLE

Syntax

```
dsfloat DS5001_TIME2ANGLE(long time)
```

Include file

`ds5001.h`

Purpose

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

Description

You need this macro for the `ds5001_read_contig` or the `ds5001_read_overl` function if the DS5001 is used in angle-based mode with a base timer cycle from 0 … 360°, set by `ds5001_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.

```
ds5001_read_overl(DS5001_1_BASE, 1, &count, len, &state, &time);
angle = DS5001_TIME2ANGLE(time);
```

Related topics

References

ds5001_read_contig........40
ds5001_read_overl........43
ds5001_set_rpm........56
DS5001_TIME2ANGLE2........47

DS5001_TIME2ANGLE2

Syntax

```
dsfloat DS5001_TIME2ANGLE2(long time)
```

Include file

`ds5001.h`

Purpose

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

Description

You need this macro for the `ds5001_read_contig` or the `ds5001_read_overl` function if the DS5001 is used in angle-based mode with a base timer cycle from 0 … 720°, set by `ds5001_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.

```
ds5001_read_overl(DS5001_1_BASE, 1, &count, len, &state, &time);
angle = DS5001_TIME2ANGLE2(time);
```

Related topics

References

ds5001_read_contig.. 40
ds5001_read_overl.. 43
ds5001_set_rpm2.. 57
DS5001_TIME2ANGLE.. 47

DS5001_TIME2FLOAT

Syntax

```
dsfloat DS5001_TIME2FLOAT(long time)
```

Include file

`ds5001.h`

Purpose

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

Description

With this function, you can convert the difference between two time stamps, that were measured by using the `ds5001_read_contig` or `ds5001_read_overl` function, to a time difference in seconds. It can be used in time-based mode.

Parameters

time Specifies the time stamp difference 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.

```
ds5001_read_overl(DS5001_1_BASE, 1, &count, len, state, time);
time_delta = DS5001_TIME2FLOAT(time[0] - time[1]);
```

Related topics

References

ds5001_read_contig.. 40
ds5001_read_overl.. 43
DS5001_TIME2FREQ.. 49

DS5001_TIME2FREQ

Syntax

```
dsfloat DS5001_TIME2FREQ(long time)
```

Include file

`ds5001.h`

Purpose

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

Description

With this function, you can calculate the frequency of a time stamp difference, which has been calculated from time stamps read before by using `ds5001_read_contig` or `ds5001_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.

```
ds5001_read_overl(DS5001_1_BASE, 1, &count, len, state, time);
freq = DS5001_TIME2FREQ(time[0] - time[1]);
```

Related topics

References

ds5001_read_contig	40
ds5001_read_overl	43
DS5001_TIME2FLOAT	48

Counting Events

Introduction

When measuring high-frequency signals the event buffer will overflow very soon. Therefore, the DS5001 provides three 32-bit event counters.

Where to go from here

Information in this section

Information in other sections

Counting Events (DS5001 Features 📖)
The DS5001 provides three 32-bit counters that can be used to count rising and/or falling edges on up to three input channels.

ds5001_counter_init

Syntax

```
void ds5001_counter_init(
   phs_addr_t base,
   Int32 counter,
   Int32 channel)
```

Include file

`ds5001.h`

Purpose

To initialize the specified event counter.

Description

This function can be used to initialize one of 3 event counters. Each counter can count rising and/or falling edges of one of the 16 DS5001 input channels.

After initialization the counter value is set to zero.

Note

The edge detection must be enabled by a preceeding call to `ds5001_read_init`, `ds5001_f2d_init` or `ds5001_pwm2d_init`.

I/O mapping

For information on the I/O mapping, refer to Counting Events on page 51.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

counter Specifies the event counter. You can use the following predefined symbols:

Predefined Symbol	Meaning
`DS5001_COUNTER_A`	Initializes counter A
`DS5001_COUNTER_B`	Initializes counter B
`DS5001_COUNTER_C`	Initializes counter C

channel Specifies the input channel within the range 1 … 16 from that the event counter will count the edges.

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_counter_read.. 52

ds5001_counter_read

Syntax

```
void ds5001_counter_read(
   phs_addr_t base,
   Int32 counter,
   Int32 mode,
   UInt32 *value)
```

Include file

`ds5001.h`

Purpose

To read the event number from the specified event counter.

Note

- The counter must have been initialized by a preceeding call to `ds5001_counter_init`.
- The edge detection must be enabled by a preceeding call to `ds5001_read_init`, `ds5001_f2d_init` or `ds5001_pwm2d_init`.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

counter Specifies the event counter to be read. You can use the following predefined symbols:

Predefined Symbol	Meaning
`DS5001_COUNTER_A`	Event counter A will be read
`DS5001_COUNTER_B`	Event counter B will be read
`DS5001_COUNTER_C`	Event counter C will be read

mode Specifies the read mode. You can use the following predefined symbols:

Predefined Symbol	Meaning
`DS5001_NO_RESET`	The counter value remains unchanged after the read access.
`DS5001_RESET`	The counter value is cleared after the read access.

value Specifies the pointer to the buffer where the counter value in the range `0x00000000` ... `0xFFFFFFFF` is written to.

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_counter_init..51
ds5001_f2d_init..20
ds5001_pwm2d_init..12
ds5001_read_init..39

Angle-Based Mode

Introduction

To set the time base to angle-based mode with an angle width of 360° or 720°.

Where to go from here

Information in this section

Example of Using Angle-Based Functions

Introduction

The following example can be used to analyze the ignition pulses of an 8-cylinder motor.

The main function contains the initialization functions for the processor board (`init`), the DS5001 (`ds5001_init`), and the event data capture unit (`ds5001_read_init`). The time base is set to angle-based mode with an angle width of 720° (`set_rpm2`) and an initialization value for the engine speed of 10,000 rpm. The actual engine speed is measured by the user-defined `measure_rpm` function, which will be executed within the `isr_t1` timer interrupt service routine. To measure the exact positions of the ignition pulses, the time base counter must be synchronized with the engine position. To do this, the user-defined application must reset the time base counter to 0 if the crankshaft position is 0°. At this crankshaft position an PHS-bus interrupt must be specified that triggers the `isr_0_degree` interrupt service routine for calculating the 8 ignition pulse positions.

Note

The following source code is not complete. Your application must contain further commands for reading the engine speed, synchronizing the time base and generating the PHS-bus interrupt.

```
#include "brtenv.h"         /* basic real time environment */
#include "ds5001.h"         /* DS5001 constants and macros */
```

```
/*************************
  global variables
**************************/
dsfloat rpm = 10000;                    /* initial values */
/* scaling value for converting time stamps to time */
dsfloat scale;
long state[10];             /* data array for input data */
long time[10];              /* data array for input data */
dsfloat a_4[8] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
dsfloat delay = 0.0
long count = 9;
/*********************************************************
   timer interrupt service routine
*********************************************************/
void isr_t1()        /* timer1 interrupt service routine */
{
   …
   /* measure RPM of engine with a user-defined function */
   rpm = measure_rpm();
   /* set new rpm value for time base */
   scale = ds5001_set_rpm2(DS5001_1_BASE, rpm);
}
/*********************************************************
   0 degree interrupt service routine
*********************************************************/
void isr_0_degree()
{
   long len;
   /* read 9 events and evaluate the 8 newest of them */
   ds5001_read_overl(DS5001_1_BASE, 1, count, &len, state, time);
   if(len == 9)
   {
      /* calculate the angle positions of the 8 ignition pulses */
      a_4[0] = DS5001_TIME2ANGLE2(time[7]);
      a_4[1] = DS5001_TIME2ANGLE2(time[6]);
      a_4[2] = DS5001_TIME2ANGLE2(time[5]);
      a_4[3] = DS5001_TIME2ANGLE2(time[4]);
      a_4[4] = DS5001_TIME2ANGLE2(time[3]);
      a_4[5] = DS5001_TIME2ANGLE2(time[2]);
      a_4[6] = DS5001_TIME2ANGLE2(time[1]);
      a_4[7] = DS5001_TIME2ANGLE2(time[0]);
      /* calculate time delay in seconds between 1. and 2. ignition pulse */
      delay = (time[6] - time[7]) * scale;
   }
}
void main()
{
   /* basic hardware initialization */
   init();
   /* initialize DS5001 board */
   err = ds5001_init(DS5001_1_BASE);
   msg_info_set(MSG_SM_RTLIB, 0, "System started.");
   ds5001_read_init(DS5001_1_BASE, 1, DS5001_RISING, 1.4, 0);
   /* start time base */
   scale = ds5001_set_rpm2(DS5001_1_BASE, rpm);
   /* initialize 0 degree interrupt */
   /* … insert your own code here */
   install_phs_int_vector(DS????_1_BASE, ?, isr_0_degree);
   /* initialize sampling clock timer */
   RTLIB_SRT_START(0.001, isr_t1);
   RTLIB_INT_ENABLE();
```

```
    while(1)
    {
        RTLIB_BACKGROUND_SERVICE();
    }
}
```

Related topics

References

ds5001_init 9
ds5001_read_init 39
ds5001_read_overl 43
ds5001_set_rpm2 57
DS5001_TIME2ANGLE2 47

ds5001_set_rpm

Syntax

```
dsfloat ds5001_set_rpm(
    phs_addr_t base,
    dsfloat rpm)
```

Include file

`ds5001.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 `0x7FFFFFFF`) is performed within 1/rpm minutes, thus representing an angle from 0 … 360°.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

rpm Specifies the speed of the time base within the range 0.00 … 9,374,998.88. The resolution is about 1.12 rpm.

If you specify 1.1175 for the `rpm` parameter, the time base accumulator is reset to normal mode (increment = 1).

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 `ds5001_read_contig` function.

Example

This example shows how to use this function.

```
dsfloat scale, rpm = 50000;
…
/* start time base */
scale = ds5001_set_rpm(DS5001_1_BASE , rpm);
…
```

Related topics

References

- Base Address of the I/O Board 7
- ds5001_init 9
- ds5001_read_contig 40
- ds5001_set_rpm2 57

ds5001_set_rpm2

Syntax

```
dsfloat ds5001_set_rpm2(
   phs_addr_t base,
   dsfloat rpm)
```

Include file

`ds5001.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 `0x7FFFFFFF`) is performed within 2/rpm minutes, thus representing an angle from 0 … 720°.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

rpm Specifies the speed of the time base within the range 0.00 … 18,749,997.76. The resolution is about 2.24 rpm.

If you specify 2.23517 for the `rpm` parameter, the time base accumulator is reset to normal mode (increment = 1).

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 `ds5001_read_contig` function.

Example

This example shows how to use this function.

```
dsfloat scale, rpm = 50000;
…
/* start time base */
scale = ds5001_set_rpm2(DS5001_1_BASE , rpm);
…
```

Related topics

References

ds5001_apu_velocity_write	62
ds5001_init	9
ds5001_read_contig	40
ds5001_set_rpm	56

Time Base Distribution

Introduction

You can use the time-base connector to distribute the time base of one DS5001 to other DS5001, DS4002, DS2210 or DS2211 boards.

Note

Time-base distribution can be done only for board revision DS5001-06 and higher. Lower board revisions do not have a time-base connector.

Where to go from here

Information in this section

Information in other sections

Implementing the Angle-Based Mode and Time-Base Distribution (DS5001 Features 🕮)
DS5001 boards of revision DS5001-06 and higher allow you to operate in angle-based mode synchronously on one or more DS5001 boards.

ds5001_apu_master_detect

Syntax

```
int ds5001_apu_master_detect(phs_addr_t base)
```

Include file

`ds5001.h`

Purpose

To detect a DS5001 or DS4002, connected to the time-base connector, which is initialized as master.

> **Note**
>
> - This function can be used only for board revision DS5001-06 and higher.
> - This function must not be used in conjunction with a connected DS2210, since this board does not support the detection of the master.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

Return value

Returns the status of the master detection. The following symbols are predefined:

Symbol	Meaning
`DS5001_MASTER_FOUND`	There is a master connected to the time-base bus.
`DS5001_NO_MASTER_FOUND`	There is no master connected to the time-base bus.

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds5001_apu_master_detect(??): Board not initialized!	The DS5001 has not been initialized by a preceding call to the `ds5001_init` function.
-916	Error	ds5001_apu_master_detect(??): DS5001 board revision 6 or higher required!	The current DS5001 board has a revision number less than 6. The functions of the time-base connector can be used only for board revision DS5001-06 and higher.

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_apu_mode_set.. 61

ds5001_apu_mode_set

Syntax

```
void ds5001_apu_mode_set(
   phs_addr_t base,
   long mode)
```

Include file

`ds5001.h`

Purpose

To specify the DS5001 as time-base bus master or slave.

Description

In the master mode the DS5001 will calculate the engine position and supplies the result to the time-base connector, from which slaves (DS5001, DS4002 or DS2210 in slave mode) can read it. The internal time base of the DS5001 is selected and the increment register is cleared. The time base 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 DS5001-06 and higher.

Parameters

base Specifies the PHS-bus base address, refer to DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

mode Specifies the mode. The following symbols are predefined:

Symbol	Meaning
`DS5001_SLAVE`	Slave mode
`DS5001_MASTER`	Master mode

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-50	Error	ds5001_apu_mode_set(??): Board not initialized!	The DS5001 has not been initialized by a preceding call to the `ds5001_init` function.
-916	Error	ds5001_apu_mode_set(??): DS5001 board revision 6 or higher required!	The current DS5001 board has a revision number less than 6. The functions of the time-base connector can be used only for board revision DS5001-06 and higher.

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_apu_master_detect........ 59
ds5001_apu_start........ 63

ds5001_apu_velocity_write

Syntax

```
void ds5001_apu_velocity_write(
   phs_addr_t base,
   dsfloat vel)
```

Include file

`ds5001.h`

Purpose

To update the angle velocity.

For further information, refer to Measuring Angle-Based Signals (DS5001 Features 🕮).

> **Note**
>
> This function can be used only for board revision DS5001-06 and higher.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

vel Specifies the angle velocity within the range 0 … 1,963,495.17 rad/s.

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_apu_mode_set 61
ds5001_apu_start 63

ds5001_apu_start

Syntax

```
void ds5001_apu_start(phs_addr_t base)
```

Include file

`ds5001.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 counter. For further information, refer to Implementing the Angle-Based Mode and Time-Base Distribution (DS5001 Features 🕮).

Note

- This function can be used only for board revision DS5001-06 and higher.
- Before you can call this function, you must set the DS5001 to master mode using `ds5001_apu_mode_set`.
- The engine position phase accumulation needs an initial value for the angle velocity. You can specify it using `ds5001_apu_velocity_write`.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
-916	Error	ds5001_apu_start(??): DS5001 board revision 6 or higher required!	The current DS5001 board has a revision number less than 6. The functions of the time-base connector can be used only for board revision DS5001-06 and higher.
-917	Error	ds5001_apu_start(??): board is not in APU master mode!	The DS5001 has not been specified as master. Use `ds5001_apu_mode_set` to specify the DS5001 as master.

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

Basics

Implementing the Angle-Based Mode and Time-Base Distribution (DS5001 Features 🕮)

References

ds5001_apu_position_read

Syntax

```
void ds5001_apu_position_read(
   phs_addr_t base,
   dsfloat *pos)
```

Include file

`ds5001.h`

Purpose

To read the current engine position.

Note

This function can be used only for board revision DS5001-06 and higher.

Parameters

base Specifies the PHS-bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

pos Specifies the pointer to the current engine position value. It is measured in rad within the range 0 … 4π.

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

ds5001_apu_position_write.. 65

ds5001_apu_position_write

Syntax

```
void ds5001_apu_position_write(
   phs_addr_t base,
   dsfloat pos)
```

Include file

`ds5001.h`

Purpose

To write the current engine position.

Description

This function writes a new engine position value to the time-base counter. Only the 23 most significant bits of the 31-bit counter are written, the remaining 8 bits are set to 0. The time-base bus master exports only the 13 most significant bits of the time-base counter. For further information, refer to Implementing the Angle-Based Mode and Time-Base Distribution (DS5001 Features 🕮).

Note

This function can be used only for board revision DS5001-06 and higher.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

pos Specifies the engine position value to be written within the range 0 … 4π. Specifying 0 will clear the engine position.

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

Base Address of the I/O Board	7
ds5001_apu_position_read	64

ds5001_apu_stop

Syntax

```
void ds5001_apu_stop(phs_addr_t base)
```

Include file

`ds5001.h`

Purpose

To stop the time-base distribution.

Description

This function stops the engine phase accumulation of the time-base counter.

> **Note**
>
> This function can be used only for board revision DS5001-06 and higher.

Parameters

base Specifies the PHS-bus base address. Refer to Base Address of the I/O Board on page 7.

Return value

None

Execution times

For information, refer to Function Execution Times on page 71.

Related topics

References

Base Address of the I/O Board...7
ds5001_apu_start...63

Bit I/O

Introduction

The following function provides a bit I/O access to the timing I/O unit.

Where to go from here

Information in this section

Information in other sections

Bit I/O (DS5001 Features 📖)
The timing I/O unit allows you to read the state of single input channels.

ds5001_bit_in

Syntax

```
int ds5001_bit_in(
   phs_addr_t base,
   int channel,
   long *state)
```

Include file

`ds5001.h`

Purpose

To read the state of an input channel.

I/O mapping

For details on the I/O mapping, refer to Bit I/O (DS5001 Features 📖).

Parameters

base Specifies the PHS bus base address, see DSxxxx_n_BASE Macros (refer to DSxxxx_n_BASE Macros on page 7).

channel Specifies the channel number within the range 1 … 16.

state Returns the state of the input channel.

Value	Meaning
0	Current state of the input channel is below the trigger level.

Value	Meaning
1	Current state of the input channel is above the trigger level.

Return value

This function returns always `DS5001_NO_ERROR`. It is only kept for compatibility reasons.

Execution times

For information, refer to Function Execution Times on page 71.

Example

This example shows how to use the function:

```
…
ds5001_bit_in(DS5001_1_BASE, 1, &state);
if (state == 1)
   …
else
   …
```

In the above example the DS5001's first channel is used for bit input.

Function Execution Times

Introduction

This section gives you basic information on the test environment and contains the mean function execution times.

Where to go from here

Information in this section

Information on the Test Environment

Test environment

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

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

The test programs that are used to measure the execution time of the functions listed below have been generated and compiled with the default settings of the `down<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

Execution times

Execution times are available for the following RTLib units:

- Initialization on page 72
- Timing I/O unit on page 72
- Incremental encoder measurement on page 73

Note

The following execution times contain mean values for a sequence of I/O accesses. The execution time of a single call might be lower because of buffered I/O access.

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
`ds5001_init`	80.81 µs	75,28 µs

Timing I/O unit

The following execution times have been measured for the signal measurement functions:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
PWM Signal Measurement		
`ds5001_pwm2d_init`	4.01 µs	3.98 µs
`ds5001_pwm2d_contig`	3.929 + c[1] · 1.160 µs	3.91 + c[1] · 1.144 µs
`ds5001_pwm2d_overl`	2.760 + c[1] · 1.160 µs	2.74 + c[1] · 1.144 µs
Square-Wave Signal Measurement		
`ds5001_f2d_init`	4.00 µs	3.98 µs
`ds5001_f2d_contig`	4.70 µs	4.68 µs
`ds5001_f2d_overl`	4.11 µs	4.07 µs

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
Phase-Shift Measurement		
`ds5001_phase_init`	5.75 µs	5.70 µs
`ds5001_phase_overl`	5.670 + c[1] · 1.924 µs	5.644 + c[1] · 1.923 µs
Incremental Encoder Measurement (see below)		
Event Data Capture		
`ds5001_read_init`	4.00 µs	3.98 µs
`ds5001_read_contig`	3.328 + c[1] · 0.572 µs	3.322 + c[1] · 0.569 µs
`ds5001_read_overl`	2.158 + c[1] · 0.572 µs	2.134 + c[1] · 0.569 µs
`ds5001_timebase_read`	1.51 µs	1.52 µs
Counting Events		
`ds5001_counter_init`	2.35 µs	2.32 µs
`ds5001_counter_read`	0.80 µs	0.79 µs
Time Base Distribution		
`ds5001_apu_start`	1.30 µs	1.30 µs
`ds5001_apu_stop`	0.72 µs	0.73 µs
`ds5001_apu_mode_set`	1.51 µs	1.50 µs
`ds5001_apu_master_detect`	0.59 µs	0.59 µs
`ds5001_apu_velocity_write`	0.73 µs	0.73 µs
`ds5001_apu_position_write`	2.09 µs	2.07 µs
`ds5001_apu_position_read`	0.81 µs	0.79 µs
Bit I/O		
`ds5001_bit_in`	1.51 µs	1.57 µs

[1] c is the number of events/periods to be evaluated

Incremental encoder measurement

The following execution times have been measured for the incremental encoder measurement functions.

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`ds5001_enc_init`	6.93 µs	6.87 µs
`ds5001_enc`:		
10 lines	28.76 µs	–
100 lines	136.34 µs	–
`ds5001_enc_clr`	0.90 µs	0.94 µs

Related topics

Basics

Information on the Test Environment..71