DS4002 Timing and Digital I/O Board

# Features

Release 2021-A – May 2021

dSPACE

## How to Contact dSPACE


| | |
|---|---|
| Mail: | dSPACE GmbH<br>Rathenaustraße 26<br>33102 Paderborn<br>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 Document

**Introduction**

This document provides feature-oriented access to the information you need to implement the functions of the DS4002.

**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.

# Introduction to the Features of the DS4002

**Basics**

The DS4002 provides a 32-bit digital I/O unit that you can use to observe input lines (switches, sensors) or control output lines (relays, displays). In addition, the board provides 8 channels for either capturing digital signals or generating arbitrary pulse patterns.

**Where to go from here**

Information in this section

DS4002 Architecture..... 10
Presenting you an overview of the functional units and architecture of the DS4002.

Feature Overview..... 10
The DS4002 Timing and Digital I/O Board provides several features.

DS4002 Interfaces..... 11
The DS4002 has interfaces for connection to a PHS-bus-based system and external devices.

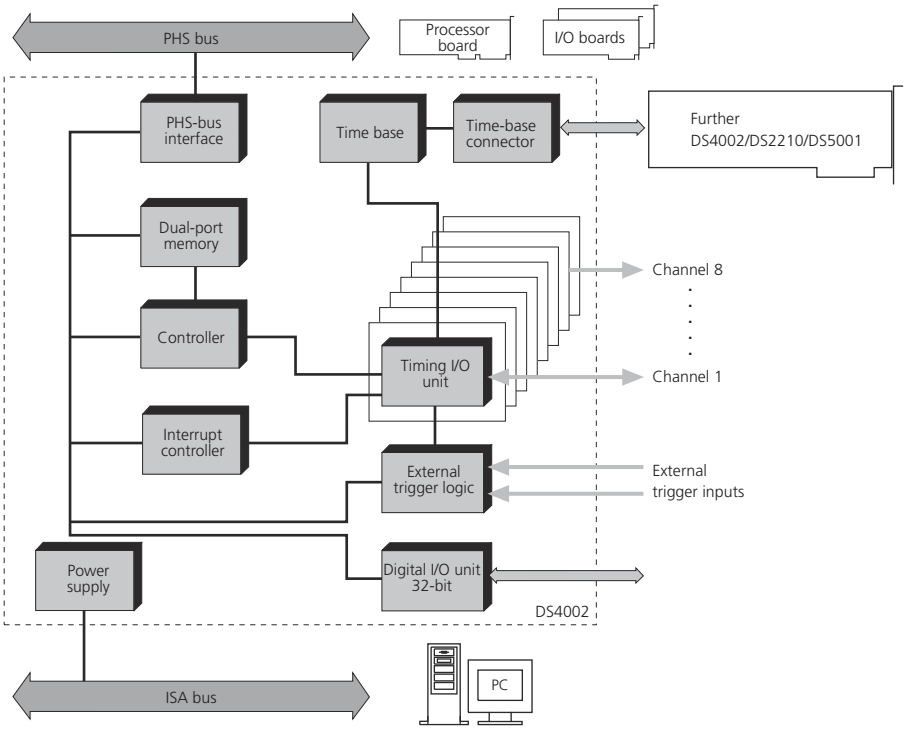
Information in other sections

Data Sheets (PHS Bus System Hardware Reference 🕮)
Summarizes the technical specifications of the hardware components.

# DS4002 Architecture

**Introduction**

The following illustration gives an overview of the functional units and architecture of the DS4002:




**Related topics**

Basics

DS4002 Interfaces ..... 11
Feature Overview ..... 10

# Feature Overview

**Features**

The DS4002 Timing and Digital I/O Board provides the following features:

**Digital I/O unit** Provides 32-bit digital I/O, refer to Digital I/O Unit on page 13.

**Timing I/O unit** Provides 8 channels to generate or measure digital signals. The unit uses a time base with 200 ns resolution, refer to Timing I/O Unit on page 19.

**Note**

The time-base connector, which is part of the timing I/O unit, is supported only for specific board versions. For details, refer to DS4002 Board Revision on page 90.

**Interrupt control** Provides various hardware interrupts, refer to Interrupts Provided by the DS4002 on page 79.

**Memory** Comprising a dual-port memory, refer to Memory Features on page 83.

**Limitations**

There are some limitations when you work with the DS4002. For details, refer to Limitations on page 89.

**Related topics**

Basics



# DS4002 Interfaces

**Introduction**

The DS4002 has interfaces for connection to a PHS-bus-based system and external devices.

**Integration into a PHS-bus-based system**

To use the DS4002, it must be integrated into a PHS-bus-based system. While the DS4002 performs the required I/O tasks, the processor board takes over the calculation of the real-time model. That is, applications using DS4002 I/O features are implemented on the processor board.

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

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

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

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

## Connection to external devices

There are two different ways to connect external devices to the DS4002. To access the I/O units of the board, connect external devices

- to the 50-pin, female I/O connector P2 of the DS4002, or
- to the optional connector panel CP4002 or the optional combined connector/LED panel CLP4002, which provides an additional array of LEDs indicating the states of the digital signals.

## Related topics

Basics

DS4002 Architecture ..... 10
Feature Overview ..... 10

# Digital I/O Unit

**Where to go from here**

Information in this section



## Basics of the Digital I/O Unit

**Characteristics**

The DS4002 controls a digital I/O unit with the following characteristics:

- 32-bit digital I/O
- Input/output mode:
  - Bits 0 … 23: mode selectable for each of the 8-bit groups (bits 0 … 7, 8 … 15, 16 … 23) individually
  - Bits 24 … 27: always in output mode
  - Bits 28 … 31: always in input mode
- TTL voltage range for 32-bit digital I/O and handshake lines

**Power-up state**

On power-up of the DS4002, all digital I/O lines except for bits 24 … 27 are in input mode. They are set to high level.

Bits 24 … 27 are always in output mode. They are set to low level on power-up of the board.

**Specifying initialization and termination values**

For the digital I/O lines that you configure as digital outputs and for the digital I/O lines 24 … 27, which are always in output mode, you can specify the digital output at the simulation start.

You can also specify the digital output at the simulation end. This allows you to drive the connected hardware into a safe final condition.

**Direct and strobed input mode**

If configured as digital inputs, the three 8-bit groups (bits 0 … 7, 8 … 15, 16 … 23) can be driven in two input modes:

- In the *direct input mode*, new data is input immediately. No additional strobe signal is necessary in this mode.
- In the *strobed input mode*, new data is latched into the digital I/O unit after a strobe signal at the PSTB line. For details, see Strobing Inputs on page 16.

**RTI/RTLib support**

You can access the digital I/O unit via DS4002 Blockset and RTLib. For details, refer to

- RTI: Digital I/O Unit (DS4002 RTI Reference 🕮)
- RTLib: Digital I/O Unit (DS4002 RTLib Reference 🕮)

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the digital I/O unit and the corresponding handshake lines, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI blocks and RTLib functions and the corresponding pins used by the digital I/O unit:

| Related RTI Block | Bit (RTI) | Related RTLib Functions | Bit (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002IN/ | Bit 0 (gr 1) | See Digital I/O Unit (DS4002 RTLib Reference 🕮) | Bit 0 | P2 34 | CP11 1 | IO0 |
| | Bit 1 (gr 1) | | Bit 1 | P2 18 | CP11 18 | IO1 |

| Related RTI Block | Bit (RTI) | Related RTLib Functions | Bit (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002OUT | Bit 2 (gr 1) | | Bit 2 | P2 2 | CP11 2 | IO2 |
| | Bit 3 (gr 1) | | Bit 3 | P2 35 | CP11 19 | IO3 |
| | Bit 4 (gr 1) | | Bit 4 | P2 19 | CP11 3 | IO4 |
| | Bit 5 (gr 1) | | Bit 5 | P2 3 | CP11 20 | IO5 |
| | Bit 6 (gr 1) | | Bit 6 | P2 36 | CP11 4 | IO6 |
| | Bit 7 (gr 1) | | Bit 7 | P2 4 | CP11 21 | IO7 |
| | Bit 8 (gr 2) | | Bit 8 | P2 37 | CP11 5 | IO8 |
| | Bit 9 (gr 2) | | Bit 9 | P2 21 | CP11 22 | IO9 |
| | Bit 10 (gr 2) | | Bit 10 | P2 5 | CP11 6 | IO10 |
| | Bit 11 (gr 2) | | Bit 11 | P2 38 | CP11 23 | IO11 |
| | Bit 12 (gr 2) | | Bit 12 | P2 6 | CP11 7 | IO12 |
| | Bit 13 (gr 2) | | Bit 13 | P2 39 | CP11 24 | IO13 |
| | Bit 14 (gr 2) | | Bit 14 | P2 23 | CP11 8 | IO14 |
| | Bit 15 (gr 2) | | Bit 15 | P2 7 | CP11 25 | IO15 |
| | Bit 16 (gr 3) | | Bit 16 | P2 40 | CP11 9 | IO16 |
| | Bit 17 (gr 3) | | Bit 17 | P2 8 | CP11 26 | IO17 |
| | Bit 18 (gr 3) | | Bit 18 | P2 41 | CP11 10 | IO18 |
| | Bit 19 (gr 3) | | Bit 19 | P2 25 | CP11 27 | IO19 |
| | Bit 20 (gr 3) | | Bit 20 | P2 9 | CP11 11 | IO20 |
| | Bit 21 (gr 3) | | Bit 21 | P2 42 | CP11 28 | IO21 |
| | Bit 22 (gr 3) | | Bit 22 | P2 10 | CP11 12 | IO22 |
| | Bit 23 (gr 3) | | Bit 23 | P2 43 | CP11 29 | IO23 |
| DS4002OUT_SINGLE_BIT | Bit 24 | See Digital I/O Unit (DS4002 RTLib Reference 🕮) | Bit 24 | P2 27 | CP11 13 | IO24 |
| | Bit 25 | | Bit 25 | P2 11 | CP11 30 | IO25 |
| | Bit 26 | | Bit 26 | P2 44 | CP11 14 | IO26 |
| | Bit 27 | | Bit 27 | P2 12 | CP11 31 | IO27 |
| DS4002IN_SINGLE_BIT | Bit 28 | See Digital I/O Unit (DS4002 RTLib Reference 🕮) | Bit 28 | P2 45 | CP11 15 | IO28 |
| | Bit 29 | | Bit 29 | P2 29 | CP11 32 | IO29 |
| | Bit 30 | | Bit 30 | P2 13 | CP11 16 | IO30 |
| | Bit 31 | | Bit 31 | P2 46 | CP11 33 | IO31 |

## Related topics

References

DS4002IN (DS4002 RTI Reference 🕮)
DS4002IN_SINGLE_BIT (DS4002 RTI Reference 🕮)
DS4002OUT (DS4002 RTI Reference 🕮)
DS4002OUT_SINGLE_BIT (DS4002 RTI Reference 🕮)

# Strobing Inputs

**Introduction**

The three 8-bit groups (bits 0 … 7, 8 … 15, 16 … 23) of the digital I/O unit can be used in the *strobed input mode*. In this mode, new data is latched into the digital I/O unit by the strobe signal.

**PSTB signal**

To use the strobed input mode, the strobe signal must be connected to the PSTB line. Input data is latched on the rising edge of the strobe signal.

Prior to the rising edge, the strobe signal must be low for at least 50 ns. After the rising edge, the strobe signal must be high for at least 50 ns.

**RTI/RTLib support**

You can access the digital I/O unit via DS4002 Blockset and RTLib. For details, refer to

- RTI: DS4002IN (DS4002 RTI Reference 🕮)
- RTLib: Digital I/O Unit (DS4002 RTLib Reference 🕮)

**I/O mapping**

The following table shows the mapping between the RTI block and RTLib functions and the corresponding pin used by the PSTB line:

| Related RTI Block | Bit (RTI) | Related RTLib Functions | Bit (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002IN | | See Digital I/O Unit (DS4002 RTLib Reference 🕮) | | P2 20 | CP11 35 | PSTB |

**Related topics**

Basics

Basics of the Digital I/O Unit ..... 13
Handshaking with External Devices ..... 16

# Handshaking with External Devices

**Introduction**

The DS4002 provides two output signals that acknowledge that data was written to or read from the digital I/O unit successfully.

Handshaking is not supported by RTI.

**PRDY handshake line**

When data was written to the digital I/O unit, this is indicated by a 400 ns low pulse at the PRDY handshake line.

### PACK handshake line

When data was read from the digital I/O unit, this is indicated by a 400 ns low pulse at the PACK handshake line.

### Output pins for the handshake lines

The following table shows the pins used by the handshake lines:

| Conn. Pin | Pin on CP | Signal |
|---|---|---|
| P2 24 | CP11 37 | $\overline{\text{PACK}}$ |
| P2 28 | CP11 39 | $\overline{\text{PRDY}}$ |

### Related topics

Basics

Basics of the Digital I/O Unit ..... 13
Strobing Inputs ..... 16

# Timing I/O Unit

**Introduction**

The DS4002 has a timing I/O unit that you can use to generate and measure digital signals.

**Where to go from here**

Information in this section



Information in other sections

# Basics

## Basics of the Timing I/O Unit

**Introduction**

Before you start working with the timing I/O unit of the DS4002, you should familiarize yourself with the basics.

**Note**

Knowing the basics is indispensable to understanding the Limitations on page 89.

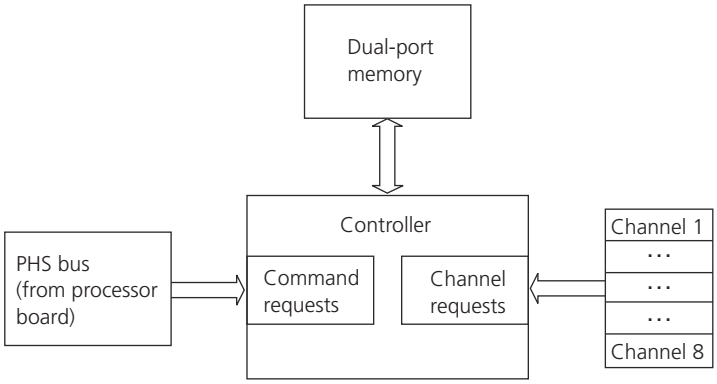
**Characteristics**

The DS4002 has a timing I/O unit that you can use to generate and measure digital signals. It has the following characteristics:

- 8 channels
- Time base with 200 ns resolution
- Each channel can individually be configured to generate or measure digital signals.
- TTL voltage range

**Controller**

The DS4002 provides a controller that is the interface between the 8 channels of the timing I/O unit and the board's dual-port memory (refer to Memory Features on page 83), which provides each channel with memory for signal generation and measurement. The controller also serves command requests from the processor board via PHS bus.




The controller works with a 200 ns time base.

### State machine code for signal generation

Every signal generated by the DS4002 can be described as a series of states – each consisting of a delay and a level change, like "after 1 ms set output high". For the channels that are in output mode, the application contains *state machine code*. The code holds all the instructions required to define and execute a series of states. In other words, the state machine code defines the signal generated on the corresponding channel. The code also contains information on interrupts to be generated and internal triggering of other channels after a state is executed.

Simple state machine code may look like this:

| State Machine Code | | | |
|---|---|---|---|
| **State #** | **Delay** | **Level** | **Interrupt** |
| 1 | 1 ms | High | Generate interrupt |
| 2 | 3 ms | Low | – |
| 3 | 2 ms | High | – |
| 4 | 5 ms | High | Generate interrupt |
| … | … | … | … |

The state machine code is executed by the *state machine* implemented in the controller.

### Channel requests

The channels of the timing I/O unit can generate or measure digital signals (output or input mode).

**Channel requests for signal generation** Channels in output mode send a *channel request* to the controller when a state is completely executed, that is, when the specified delay has expired. The controller reads the required information – such as the delay and level – for the next state to be executed from the state machine code stored in the dual-port memory, and transmits it to the channel.

**Channel requests for signal measurement** Channels in input mode send a *channel request* to the controller when a rising or falling edge was captured. The controller reads the edge direction from the channel and the corresponding time stamp from the time-base counter. Then the controller writes this information to the event-buffer in the dual-port memory.

### Command requests

If you perform signal generation with the timing I/O unit, you can modify the signal parameters during run time. You can change the frequency and the duty cycle of a PWM signal, for example.

Whenever you change one or more parameters of a signal, the connected dSPACE processor board sends a command request for run-time parameter updates to the DS4002 controller, which serves the command request.

**Processing of commands**

The controller serves the channel requests and command requests according to a priority scheme. For details, refer to Priority-based processing of requests on page 93.

**Related topics**

Basics

Generation of Digital Signals ..... 23
Measurement of Digital Signals and Event Capture ..... 54

# Generation of Digital Signals

**Introduction**

The timing I/O unit of the DS4002 can be used to generate digital signals on up to 8 channels.

**Where to go from here**

Information in this section



## PWM Signal Generation (PWM, PWM3)

**Introduction**

PWM signal generation is crucial to many motor and motion control applications. PWM signals are pulse trains with variable frequency and pulse width.

**Where to go from here**

Information in this section

# Basics of PWM Signal Generation

**Basics**

The width of the pulses changes according to a modulating signal. When a PWM signal is applied to the gate of a power transistor, it causes the turn-on/turn-off intervals of the transistor to change, according to the modulating signal. The frequency of a PWM signal is usually much higher than that of the modulating signal, so that the energy delivered to the motor and its load depends mainly on the modulating signal.

With the DS4002, you can generate 1-phase as well as 3-phase PWM signals.

**Related topics**

Basics

1-Phase PWM Signal Generation (PWM)....24
3-Phase PWM Signal Generation (PWM3)....27

# 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. The channels are not synchronized.

**Note**

When PWM signal generation starts, the initial output signal level is undefined (high or low).

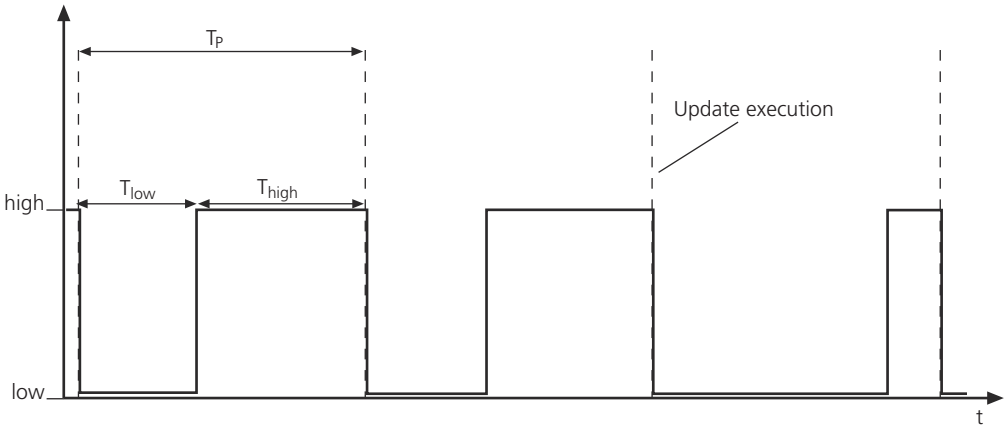
**PWM period and duty cycle**

You can specify the PWM period $T_P$ (= $T_{high}$ + $T_{low}$) for each 1-phase PWM signal individually. Depending on the number of channels of the timing I/O unit you use in your application, you can specify $T_P$ in the following range:

| Channels Used | $T_{P\_min}$ | $T_{P\_max}$ |
|---|---|---|
| 1 | 1.2 µs | 107 s |
| 8 | 8 µs | 107 s |

The minimum PWM period is determined by the processing time of the board controller. Refer to Limitations Due to the Controller Processing Time on page 91.

You can also specify the duty cycle. The following illustration shows how the duty cycle d (= $T_{high}$/ $T_P$) is defined. The available duty cycle range is 0 … 1 (0 … 100 %).




If the PWM period $T_P$ or the duty cycle d is changed during run time, the block update mode is used: New values become effective with the next PWM period, beginning with the low PWM level.

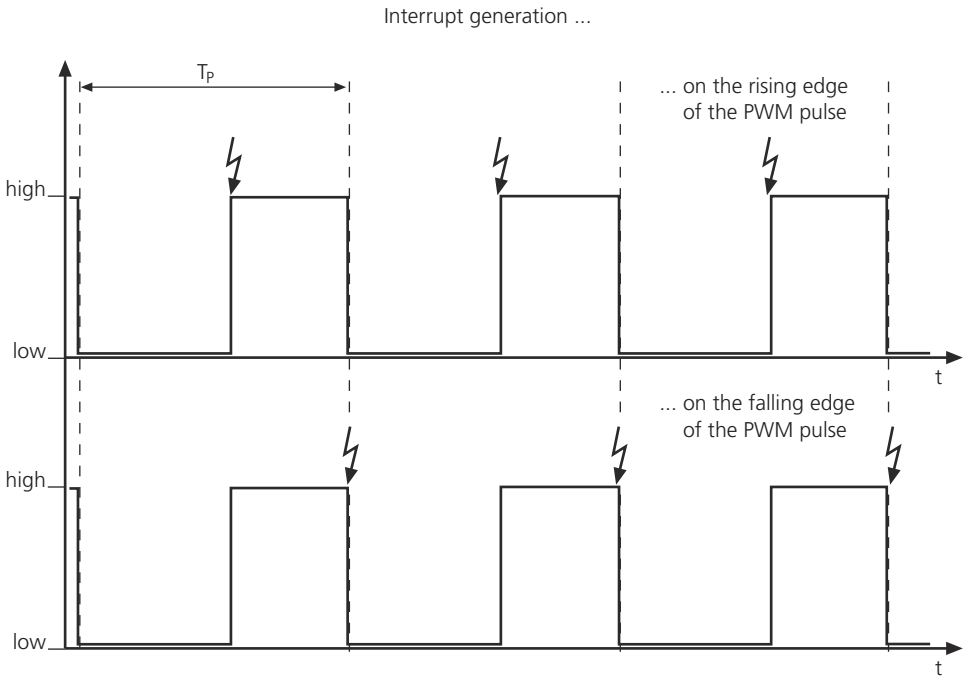
In the illustration above, an update of the duty cycle d is executed at the beginning of the third PWM period.

**Note**

Due to quantization effects, you will encounter considerable deviations between the desired PWM period $T_P$ and the generated PWM period, especially for high PWM frequencies. Refer to Quantization Effects on page 90.

**Interrupt via PWM signal generation**

When you perform 1-phase PWM signal generation on channel 1 or 2, you can enable interrupt generation for these channels. In this case, you can specify that an interrupt is generated on the rising or falling edge of the corresponding PWM pulse. See Signal Generation Interrupt on page 80 for more details.




**RTI/RTLib support**

You can perform 1-phase PWM signal generation via DS4002 Blockset and RTLib. For details, refer to

- RTI: DS4002PWM1_OUT (DS4002 RTI Reference 🕮)
- RTLib: 1-Phase PWM Signal Generation (PWM) (DS4002 RTLib Reference 🕮)

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI block and the RTLib functions and the corresponding pins used to provide 1-phase PWM signals.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002PWM1_OUT | Ch 1 | See 1-Phase PWM Signal Generation (PWM) (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit.....20

# 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.

The DS4002 timing I/O unit has 8 channels. Since 3-phase PWM signal generation requires one output channel per phase, you can implement up to two 3-phase PWM signals at the same time in an application.

**PWM period and duty cycle**

The PWM period and the duty cycle of the two 3-phase PWM signals can be specified independently. For PWM3 signals, the PWM period $T_P$ (= $T_{high}$ + $T_{low}$) applies to each of the three PWM3 phases. The maximum $T_P$ value is 107 s. The minimum $T_P$ value depends on the number of channels of the timing I/O unit you use in your application. If you implement one 3-phase PWM signal in your application, $T_{P, min}$ is 4.0 µs (without Interrupt via PWM3 signal generation on page 28) or 5.0 µs (with Interrupt via PWM3 signal generation on page 28). To calculate the minimum $T_P$ value for applications using more channels of the timing I/O unit, refer to Limitations Due to the Controller Processing Time on page 91.

For each of the three phases, you can specify the duty cycle $d_x$ (x = 1, 2, 3) individually. The duty cycle is defined as follows:

$d_x = T_{high,x} / T_P$

where $T_P$ is the PWM period. The available duty cycle range is 0 … 1 (0 … 100 %).

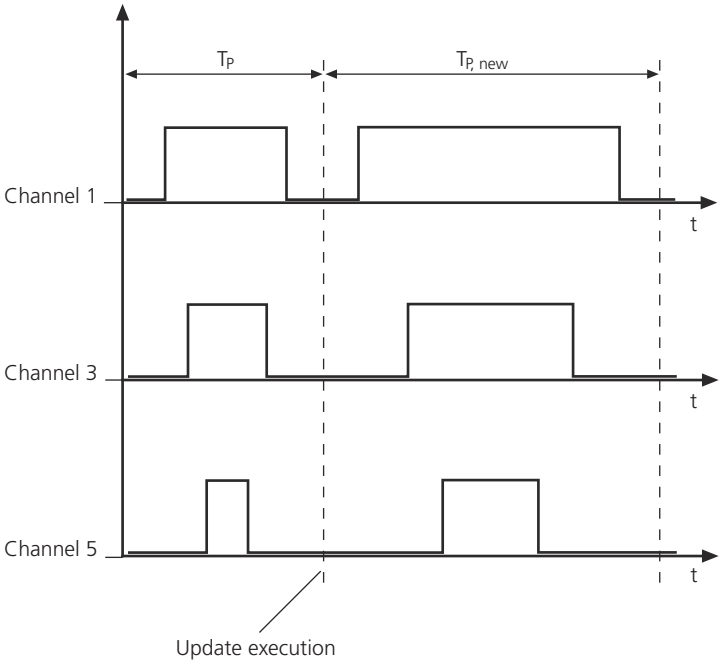
If the PWM period $T_P$ or the duty cycle d is changed during run time, the synchronous update mode is used: New values become effective synchronously for all the 3 phases with the next PWM period.

**Note**

Due to quantization effects, you will encounter considerable deviations between the desired PWM period $T_P$ and the generated PWM period, especially for high PWM frequencies. Refer to Quantization Effects on page 90.

**Pulse pattern**

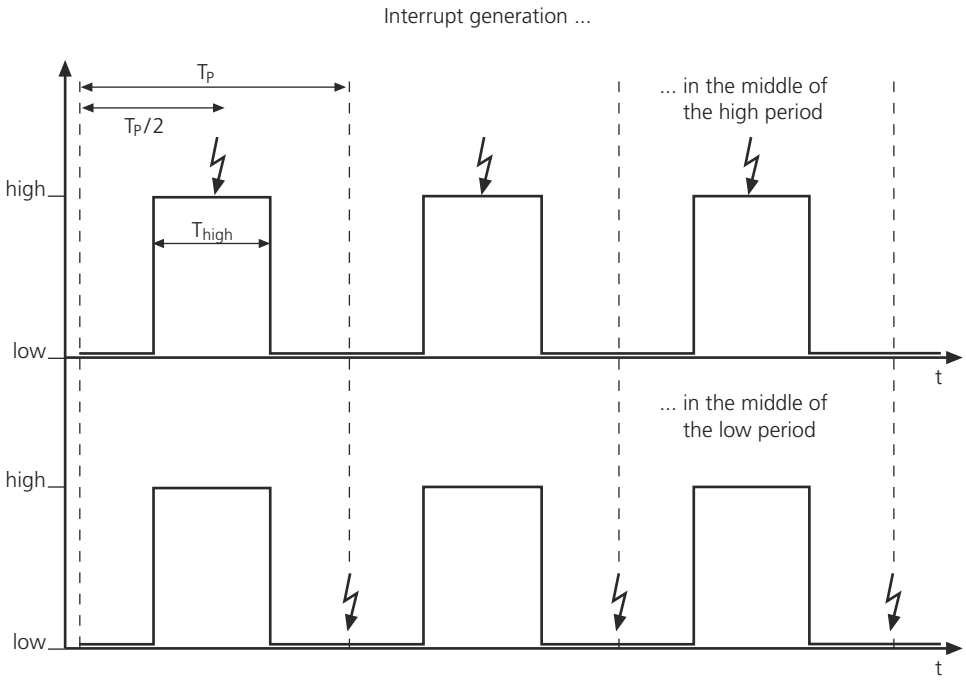
If you perform 3-phase PWM signal generation using channels 1, 3 and 5, for example, the pulse pattern of a single PWM period may look like this:




The illustration above also shows the update of the PWM period $T_P$ and the duty cycles $d_x$ during run time.

**Interrupt via PWM3 signal generation**

If you specify channel 1 or 2 as the first channel for 3-phase PWM signal generation, you can enable interrupt generation. In this case, you can specify that an interrupt is generated in the middle of the high or low period of the corresponding PWM3 signal. See Signal Generation Interrupt on page 80 for more details.




**Note**

If you generate PWM3 signals with a small PWM period $T_P$, the interrupt may not occur exactly in the middle of the high or low PWM3 period. The maximum deviation is 200 ns.

**RTI/RTLib support**

You can perform 3-phase PWM signal generation via DS4002 Blockset and RTLib. For details, refer to

- RTI: DS4002PWM3_OUT (DS4002 RTI Reference 🕮)
- RTLib: 3-Phase PWM Signal Generation (PWM3) (DS4002 RTLib Reference 🕮)

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI block and the RTLib functions and the corresponding pins used to provide 3-phase PWM signals.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002PWM3_OUT | Ch 1 | See 3-Phase PWM Signal Generation (PWM3) (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit....20
PWM Signal Generation (PWM, PWM3)....23

# Generation of Simple Signals

**Where to go from here**

Information in this section

Square-Wave Signal Generation (D2F)....31
The timing I/O unit of the DS4002 provides outputs for square-wave signal generation (D2F) on up to 8 channels.

Monoflop Signal Generation....32
The timing I/O unit of the DS4002 provides outputs for monoflop signal generation on up to 8 channels.

# 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. The channels are not synchronized.

**Note**

When D2F signal generation starts, the initial output signal level is undefined (high or low).

**Frequency range**

For square-wave signal generation on the DS4002, you can specify the frequency $f_{D2F}$ for each channel individually. Depending on the number of channels of the timing I/O unit you use in your application, you can specify $f_{D2F}$ in the following range:

| Channels Used | $f_{D2F\_min}$ | $f_{D2F\_max}$ |
|---|---|---|
| 1 | 0.01 Hz | 833.33 kHz |
| 8 | 0.01 Hz | 125 kHz |

The maximum D2F frequency is determined by the processing time of the board controller. Refer to Limitations Due to the Controller Processing Time on page 91.

If the frequency $f_{D2F}$ is changed during run time, the block update mode is used: New values become effective for the next D2F period, starting with the low D2F level.

**Note**

Due to quantization effects, you will encounter considerable deviations between the desired frequency and the generated frequency, especially for high D2F frequencies. Refer to Quantization Effects on page 90.

**Interrupt via D2F signal generation**

If you perform square-wave signal generation on channel 1 or 2, you can enable interrupt generation for these channels. In this case, you can specify that an interrupt is generated on the rising or falling edge of the corresponding D2F signal. See Signal Generation Interrupt on page 80 for more details.

**RTI/RTLib support**

You can perform square-wave signal generation via DS4002 Blockset and RTLib. For details, see

- RTI: DS4002DTOF (DS4002 RTI Reference 🕮)
- RTLib: Square-Wave Signal Generation (D2F) (DS4002 RTLib Reference 🕮)

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI block and the RTLib functions and the corresponding pins used to provide D2F signals.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002DTOF | Ch 1 | See Square-Wave Signal Generation (D2F) (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit.....20

# Monoflop Signal Generation

**Introduction**

The timing I/O unit of the DS4002 provides outputs for monoflop signal generation on up to 8 channels. Monoflop signals can be used to generate ignition pulses, for example.

After monoflop signal generation is triggered (see below), a high-active single pulse – the monoflop pulse – is output at the specified channel.

**Note**

When monoflop signal generation starts, the initial output signal level is undefined (high or low).

### Monoflop pulse length

For monoflop signal generation on the DS4002, you can specify the monoflop pulse length $T_{Mono}$ for each channel individually. Depending on the number of channels of the timing I/O unit you use in your application, you can specify $T_{Mono}$ in the following range:

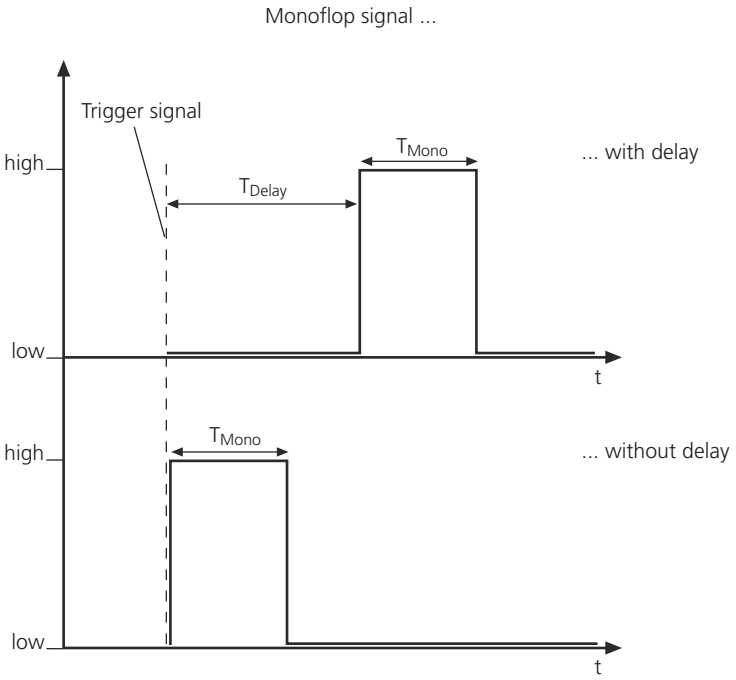
| Channels Used | $T_{Mono\_min}$ | $T_{Mono\_max}$ |
|---|---|---|
| 1 | 0.6 µs | 107 s |
| 8 | 4 µs | 107 s |

The minimum monoflop pulse length is determined by the processing time of the board controller. Refer to Limitations Due to the Controller Processing Time on page 91.

If $T_{Mono}$ is changed during run time, the new value becomes effective for the next monoflop pulse.

### Delayed pulse generation

For monoflop pulses, you can also specify a time delay, $T_{Delay}$. After monoflop signal generation is triggered, the output remains low for $T_{Delay}$. Then the monoflop pulse is output. The following illustration shows a monoflop signal with and without delay.




Depending on the number of channels of the timing I/O unit you use in your application, you can specify $T_{Delay}$ in the following range:

| Channels Used | $T_{Delay\_min}$ | $T_{Delay\_max}$ |
|---|---|---|
| 1 | 0.6 µs | 107 s |
| 8 | 4 µs | 107 s |

If $T_{Delay}$ is changed during run time, the new value becomes effective for the next delayed monoflop pulse.

**Note**

Due to quantization effects, you will encounter considerable deviations between the desired values $T_{Mono}$ and $T_{Delay}$, and the values for $T_{Mono}$ and $T_{Delay}$ of the generated monoflop signal, especially for small $T_{Mono}$ and $T_{Delay}$ values. Refer to Quantization Effects on page 90.

### Triggering monoflop signal generation

There are several ways to trigger monoflop signal generation:

- Via RTLib's `ds4002_mono_start` function (see ds4002_mono_start (DS4002 RTLib Reference 🕮))
- Via external triggering (only RTLib, see Triggering the Start of Signal Generation Externally on page 51)
- Via internal triggering (see Specifying Interrupt Generation and Internal Triggering on page 41)

**Interrupt via monoflop signal generation**

If you perform monoflop signal generation on channel 1 or 2, you can enable interrupt generation for these channels. In this case, you can specify that an interrupt is generated on the rising or falling edge of the corresponding monoflop pulse. See Signal Generation Interrupt on page 80 for more details.

**RTI/RTLib support**

You can perform monoflop signal generation via DS4002 Blockset and RTLib. For details, refer to

- DS4002MONO_Bx_Cy (DS4002 RTI Reference 🕮)
- Monoflop Signal Generation (DS4002 RTLib Reference 🕮)

**Example**

For an example of monoflop signal generation implemented with RTLib4002, refer to Example of Using the Monoflop Signal Generation Functions (DS4002 RTLib Reference 🕮).

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTLib functions and the corresponding pins used to provide monoflop signals.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002MONO_Bx_Cy | Ch 1 | See Monoflop Signal Generation (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit.....20

# Generation of Arbitrary Signals

**Introduction**

In addition to generating standard digital pulse patterns the timing I/O unit allows you to generate arbitrary digital pulse patterns on up to 8 channels. To do so, you have to program the patterns directly in C code with RTLib4002.

**Where to go from here**

Information in this section

Basics of Generating Arbitrary Signals.....37
Provides basic information on using a state machine code for generating arbitrary signals

Defining and Specifying States.....39
Each state defined in a state machine code contains information on its delay and level. A state can also hold information on internally triggering other channels, on generating interrupts, and on the program flow.

Starting Signal Generation.....45
To start arbitrary signal generation, you have to use a RTLib function.

Updating State Parameters.....46
Once defined in the state machine code, each state can be updated during run time.

Example of Generating an Arbitrary Signal.....49
The example shows the states of a state machine code, the program flow and the resulting digital signal.

Information in other sections

Basics of the Timing I/O Unit.....20
Basic information on the controller serving the timing I/O unit of the DS4002.

# Basics of Generating Arbitrary Signals

**State machine code**

Each signal can be described as a series of states – each consisting of a delay and a level change, like "after 1 ms set output level high". To generate arbitrary signals with the DS4002, you have to initialize and define a *state machine code* first. It represents the pulse pattern, and consists of a *number of states* and an optional *entry point address*:

- A state consists of a delay and a level change. It may also contain information on interrupt generation and internal triggering.
- The entry point can be used to program a loop for periodical pulse sequences, for example.

The state machine code can be loaded to one or more channels of the DS4002.

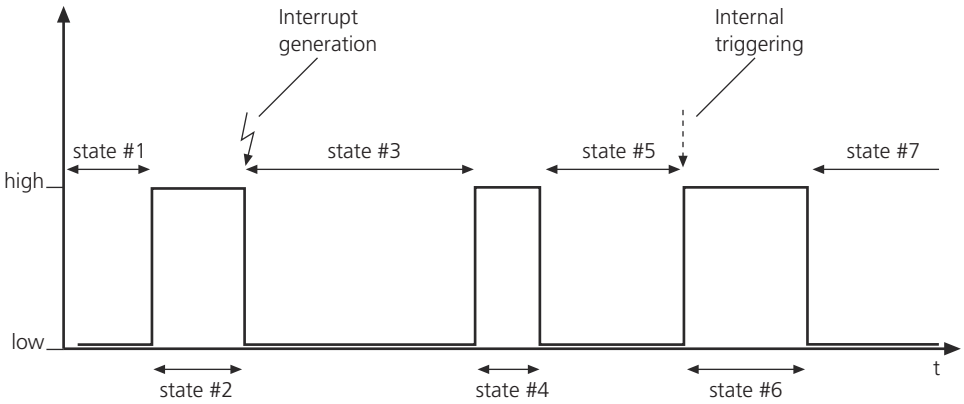
**Programming structure**

Adhere to the following structure when programming arbitrary digital pulse patterns on up to 8 channels:

| | RTLib Function | Purpose |
|---|---|---|
| **Defining the first state machine code, see Defining and Specifying States on page 39** | | |
| | `ds4002_output_init` | To reset and initialize the first state machine code. |
| | `ds4002_define_state` | To define a state of the first state machine code. |
| | … | (Further state definitions) |
| | `ds4002_define_entry` | To define an entry point in the first state machine code. |
| | `ds4002_define_state` | To define a state of the first state machine code. |
| | … | (Further state definitions) |
| | `ds4002_load_states` | To load the first state machine code to the specified channel(s). |
| … | | |
| **Defining the n-th state machine code, see Defining and Specifying States on page 39** | | |
| | `ds4002_output_init` | To reset and initialize the n-th state machine code. |
| | `ds4002_define_state` | To define the states of the n-th state machine code. |
| | … | (Further state definitions) |
| | `ds4002_define_entry` | To define an entry point in the n-th state machine code. |
| | `ds4002_define_state` | To define a state of the first state machine code. |
| | … | (Further state definitions) |
| | `ds4002_load_states` | To load the n-th state machine code to the specified channel(s). |
| **Starting signal generation, see Starting Signal Generation on page 45** | | |
| | `ds4002_start_channels` | To start arbitrary signal generation on the channels where a state machine code is loaded. |
| **Updating state parameters during run time, see Updating State Parameters on page 46** | | |
| | `ds4002_update_state` | To change the parameters of a state during run time. |

**Resulting pulse pattern**

As an example, a digital pulse pattern programmed with RTLib functions for arbitrary signal generation may look like this:




**Example**

- For an example of a state machine code and the resulting digital output signal, refer to Example of Generating an Arbitrary Signal on page 49.
- For an example based on RTLib4002, refer to Example of Using the Arbitrary Signal Generation Functions (DS4002 RTLib Reference 🕮).
- For an S-function example based on RTLib4002, refer to Example of Implementing Arbitrary Signal Generation Code as S-Function (DS4002 RTLib Reference 🕮).

**RTI/RTLib support**

Use RTLib for the DS4002 to generate arbitrary signals. For details, see Arbitrary Signal Generation (DS4002 RTLib Reference 🕮).

Using RTI, you have to program the desired signal with RTLib4002 functions, and incorporate your C code in a Simulink S-function. For details, refer to Implementing S-Functions (RTI and RTI-MP Implementation Guide 🕮).

**I/O mapping**

The following table shows the mapping between the RTLib functions and the corresponding pins used for the generation of arbitrary signals.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|
| See Arbitrary Signal Generation (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

References

Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮)

# Defining and Specifying States

**Introduction**

Each state defined in a state machine code contains information on its delay and level. A state can also hold information on internally triggering other channels, on generating interrupts, and on the program flow.

**Parameters of a state**

To define a single state, you have to specify the parameters of the ds4002_define_state function. The following parameter values are available:

| ds4002_define_state | (delay | level | trigger | instr | count) |
|---|---|---|---|---|---|
| | `DS4002_DELAY()` | `DS4002_HIGH` | `DS4002_MASK()` | `DS4002_CONTINUE` | `0` |
| | `DS4002_WAIT` | `DS4002_LOW` | `DS4002_INTERRUPT` | `DS4002_GOTO` | `0` |
| | `DS4002_ANGLE()` | | `0` | `DS4002_JUMP` | Jump address |
| | `DS4002_ANGLE2()` | | | `DS4002_LOADCOUNTER` | Counter value |
| | | | | `DS4002_REPEAT` | `0` |

For details on the parameters, refer to

- Specifying the Delay and Level on page 40
- Specifying Interrupt Generation and Internal Triggering on page 41
- Specifying the Program Flow on page 42

### Number of states

The maximum size of the state machine code – 256 program words, each word contains 32 bits – limits the number of states that can be defined.

- States using the `DS4002_LOADCOUNTER` or `DS4002_JUMP` commands (refer to Specifying the Program Flow on page 42) as the `instr` parameter require 2 words.
- All other states require 1 word in the state machine code.

Therefore, up to 256 states can be defined in a state machine code, which can be loaded to one or more channels of the DS4002.

### Updating a state

You can change the delay and level of a state during run time. Refer to Updating State Parameters on page 46.

### Specifying the Delay and Level

When you define or update the states of a digital signal via the `ds4002_define_state` or `ds4002_update_state` function, the following state parameters allow you to specify the delay and level:

| Parameter | Purpose |
|---|---|
| `delay` | To specify the delay of a state, or to wait for an external or internal trigger event via the `DS4002_WAIT` constant (see below). All the actions of a state such as level change and interrupt generation are carried out *after* the delay or trigger event. |
| `level` | To specify the output signal level – high or low – after the delay or the trigger event. |

**Delay value** Use the `DS4002_DELAY` macro to specify the `delay` value in seconds. In the example below, a state with a delay of 6 µs is defined:

```
ds4002_define_state(DS4002_DELAY(6e-6), DS4002_LOW, 0,DS4002_CONTINUE,0);
```

Keep in mind the available value range:

**Maximum delay value** This value is 107.374 s.

**Minimum delay value** This value depends on the number of channels currently used for signal generation and measurement. As a rule of thumb, the delay must be at least

600 ns + ((active_channels – 1) · 400 ns).

This allows the controller to serve all output channels of the DS4002. For details, see Limitations Due to the Controller Processing Time on page 91.

**Note**

- In angle-based mode, you have to specify an angle value instead of a delay. Refer to Generating Angle-Based Signals on page 69.
- You can also specify smaller delay values – even `delay` = 0 is possible. In this case, the corresponding channel is served as fast as possible. Some edges may be delayed, but the following level changes will occur at the correct time, since delay errors are compensated with the next delay and not accumulated.

**Waiting for a trigger** If you want signal generation to be triggered by an external or internal signal, you must specify the `DS4002_WAIT` constant as the `delay` parameter. In this case, state execution is stopped and the output signal is not changed until the trigger event.

If you use an external trigger, a constant delay of about 1 µs (with a jitter of 150 ns) occurs due to internal synchronization and processing times. Refer to Triggering the Start of Signal Generation Externally on page 51.

**Synchronous signal generation** Using the `DS4002_WAIT` constant for several channels, you can trigger the synchronous start of signal generation. To ensure that the signals generated on several channels remain synchronous, the delay of each state executed after the trigger event should be specified according to the rule of thumb for minimum delay values (see above). This allows the controller to process all the channels of the timing I/O unit.

For information on the effects of specifying too small delay values, refer to Limitations Due to the Controller Processing Time on page 91.

### Specifying Interrupt Generation and Internal Triggering

When you define the states of a digital signal via the `ds4002_define_state` function, the following state parameter allows you to specify interrupt generation and internal triggering:

| Parameter | Purpose |
|---|---|
| `trigger` | To generate an interrupt, and to internally trigger signal generation on other channels. |

The state machine code of the channel to be triggered must contain a state with the `DS4002_WAIT` constant as the `delay` parameter. Refer to Specifying the Delay and Level on page 40.

**Note**

- Interrupt generation and internal triggering are performed *after* the delay or trigger event of the corresponding state.
- Only channels 1 and 2 can generate interrupts and trigger signal generation on other channels.

### Specifying the Program Flow

When you define the states of a digital signal via the `ds4002_define_state` function, the following state parameter allows you to specify the program flow for arbitrary signal generation:

| Parameter | Purpose |
|---|---|
| `instr` | To specify which state is executed next. |

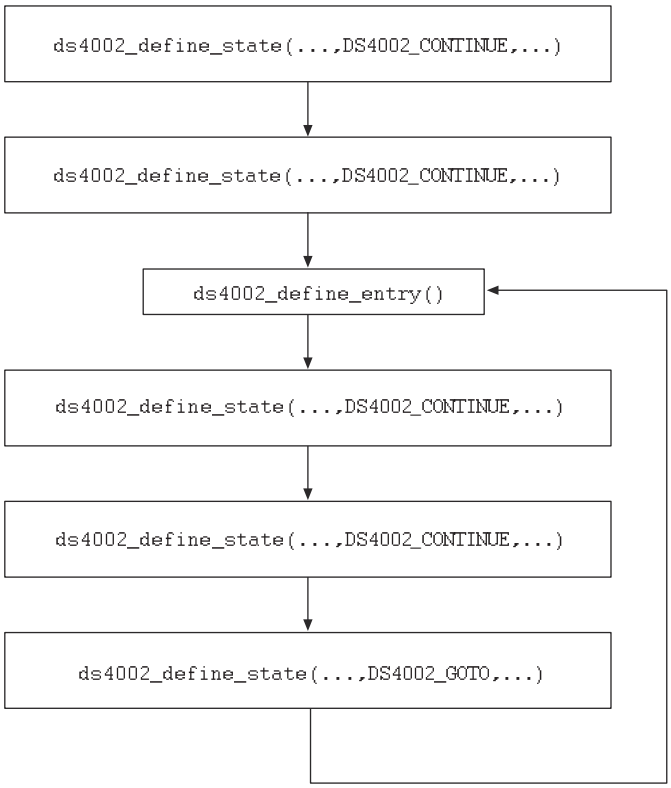
The `instr` parameter allows you to program loops to implement periodic signals, for example. Using a loop counter, you can specify the number of periods to be generated before the generation of another pulse pattern.

**Proceeding to the next state** You can use the `DS4002_CONTINUE` command as the `instr` parameter to easily proceed to the next state defined in the state machine code.

**Implementing a loop** You can use the `DS4002_JUMP` or `DS4002_GOTO` commands as the `instr` parameter to program loops within the state machine code. Infinite loops allow you to generate periodic signals.

**Proceeding to a labeled state** You can use the `DS4002_GOTO` command as the `instr` parameter to proceed to a state labeled with an entry point. The entry point has to be defined beforehand using the `ds4002_define_entry` function. In the C code, place this function directly before the state you want to proceed to.

The illustration below shows the implementation of an infinite loop:

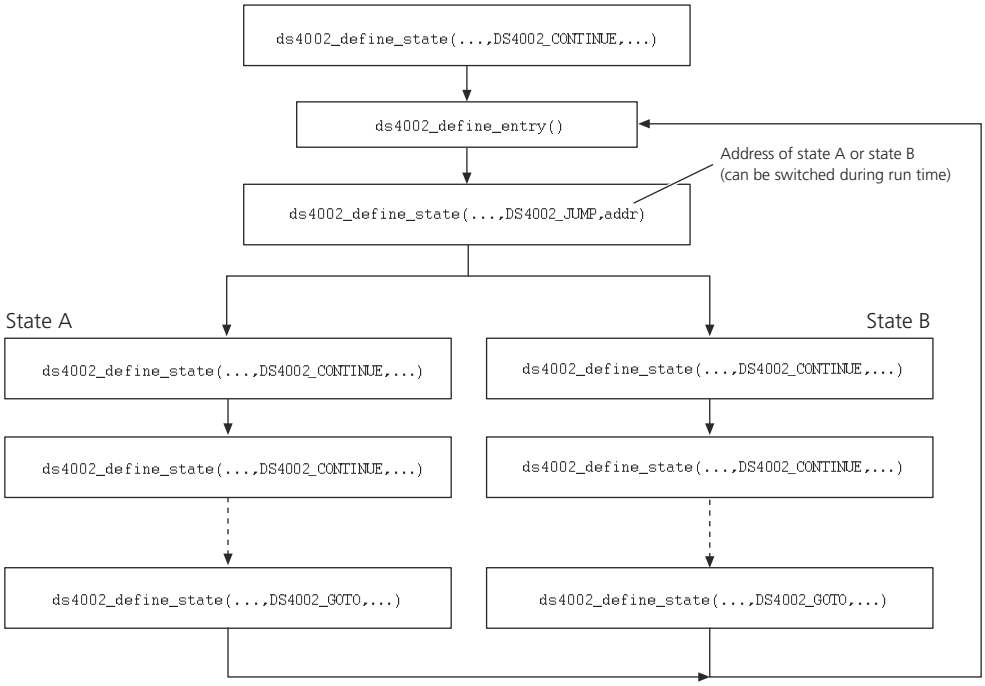


**Note**

Define only one entry point within a state machine code. If you define several entry points, the last `ds4002_define_entry` function call overwrites all preceding entry point definitions.

**Proceeding to any state** You can use the `DS4002_JUMP` command as the `instr` parameter to proceed to any state defined in the state machine code. To get the address of the state, you can use the return value of the `ds4002_define_state` function. Valid values are in the range 0 … 255.

The state address specified for the `DS4002_JUMP` command can be updated during run time. This allows you to implement a switch within the program flow. See the illustration below for an example:




**Note**

- Do not use the `DS4002_JUMP` command for a state that has the `DS4002_WAIT` constant as the `delay` parameter.
- Do not use the `DS4002_JUMP` command for a state that is defined within a counter loop, in other words, between states with the `DS4002_LOADCOUNTER` and the `DS4002_REPEAT` command.

**Implementing a counter loop** You can use the `DS4002_LOADCOUNTER` and `DS4002_REPEAT` commands as `instr` parameters to load a counter, and execute a sequence of states while the

counter value is decremented. Counter loops allow you to generate periodic signals for a finite number of periods.

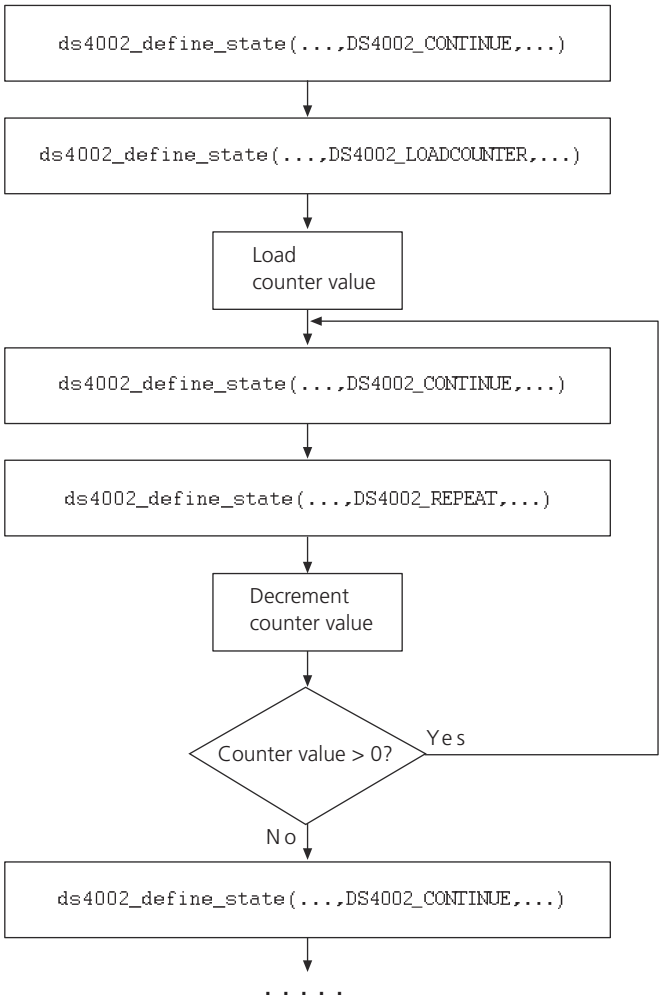
Use the `DS4002_LOADCOUNTER` command to load the counter with a value in the range 1 … 256. The program then automatically proceeds to the next state. Use the `DS4002_REPEAT` command for a state to decrement the counter by 1. Then the program proceeds to:

- the first state defined after the corresponding `DS4002_LOADCOUNTER` state if the counter value is still > 0.
- the next state defined after the `DS4002_REPEAT` state if the counter value is 0.

**Tip**

To program a loop with more than 256 periods, you can line up several counter loops – each containing a `DS4002_LOADCOUNTER/DS4002_REPEAT` construction.

The illustration below shows the implementation of a counter loop.



**Note**

- Do not use the `DS4002_LOADCOUNTER` command for a state that has the `DS4002_WAIT` constant as the `delay` parameter.
- Do not use the `DS4002_JUMP` command for a state that is defined within a counter loop, in other words, between states with the `DS4002_LOADCOUNTER` and the `DS4002_REPEAT` command.

**Related topics**

Basics

Basics of the Timing I/O Unit....20
Generation of Arbitrary Signals....36
Starting Signal Generation....45
Updating State Parameters....46

Examples

Example of Generating an Arbitrary Signal....49

References

ds4002_define_state (DS4002 RTLib Reference 🕮)

# Starting Signal Generation

**Introduction**

To start arbitrary signal generation on the channels where a state machine code is loaded, you have to use RTLib's ds4002_start_channels function.

**Note**

When you start signal generation, the output signal level for the very first state of the corresponding state machine code is undefined.

**Related topics**

Basics

Basics of the Timing I/O Unit....20
Defining and Specifying States....39
Generation of Arbitrary Signals....36

Examples

Example of Generating an Arbitrary Signal....49

References

ds4002_start_channels (DS4002 RTLib Reference 🕮)

# Updating State Parameters

**Introduction**

Once defined in the state machine code, each state can be updated during run time. This allows you to flexibly change the characteristics of the generated signal. You have to use the `ds4002_update_state` function separately for each state you want to update.

**Updateable parameters**

The following state parameters can be changed during run time:

- The delay value and the level of the corresponding state. However, you cannot change from using a `DS4002_WAIT` constant to a delay value, and vice versa. Refer to Specifying the Delay and Level on page 40.
- The state address specified for the `DS4002_JUMP` command. Refer to Implementing a loop on page 42.
- The counter loop value that is specified by the `DS4002_LOADCOUNTER` command. Refer to Implementing a counter loop on page 43.

**Note**

You cannot change the `trigger` and `instr` parameters during run time.

**Executing the update**

The `ds4002_update_state` function writes the new parameter values to the corresponding state machine code contained in a buffer of the dual-port memory. To make the buffer with the updated state machine code available for the corresponding output channel, you have to use RTLib4002's `DS4002_EXEC_CMD` macro, and specify one of the four available update modes. For parameter updates, the DS4002 internally uses the swinging buffer principle. Refer to Swinging Buffer Principle on page 84.

**Update modes**

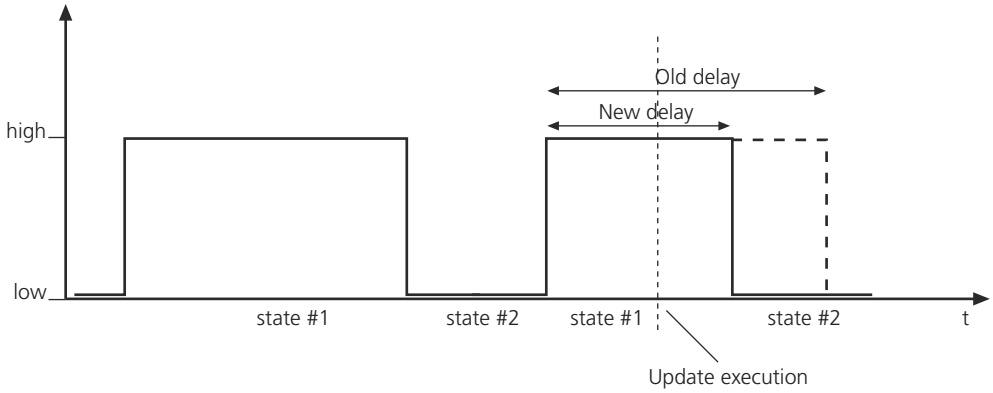
There are four different update modes:

- Immediate mode on page 47
- Next delay mode on page 47
- Block mode on page 47
- Synchronous mode on page 48

The update mode is directly related to operating mode of the three swinging buffers (refer to Swinging Buffer Principle on page 84).

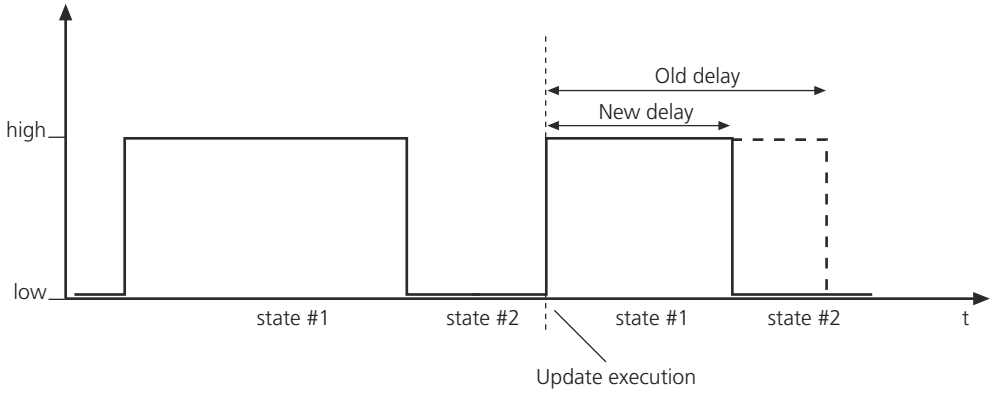
**Immediate mode** With the `DS4002_CMD_IMMEDIATE` command, you can implement the immediate mode. In this mode, the update becomes effective even for the state that is currently being executed.

The illustration below shows the immediate update of the `delay` value of state #1.




**Next delay mode** With the `DS4002_CMD_NEWDATA` command, you can implement the next delay mode. In this mode, the update becomes effective for the next state to be executed. The state currently being executed is not affected.

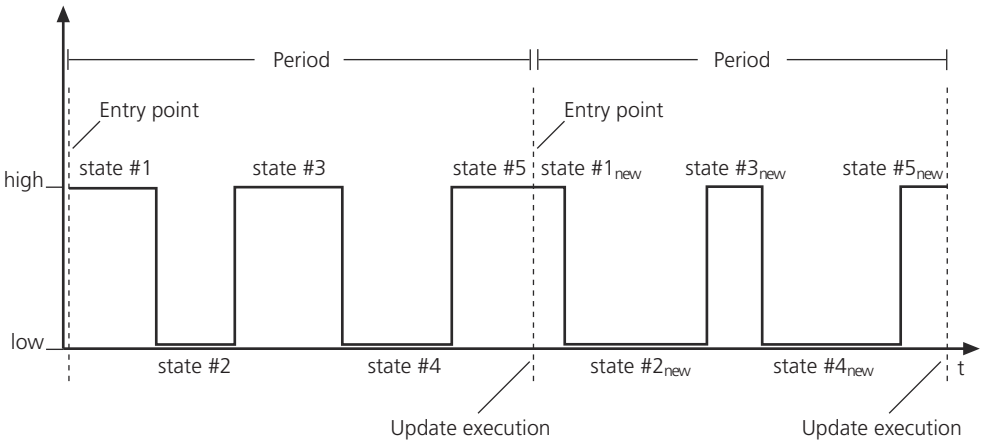
The illustration below shows the update of the `delay` value of state #1.




**Block mode** With the `DS4002_CMD_BLOCKDATA` command, you can implement the block mode. In this mode, the update becomes effective after the next `DS4002_GOTO` command is executed. Since the `DS4002_GOTO` command and an entry point can be used to implement an infinite loop (refer to

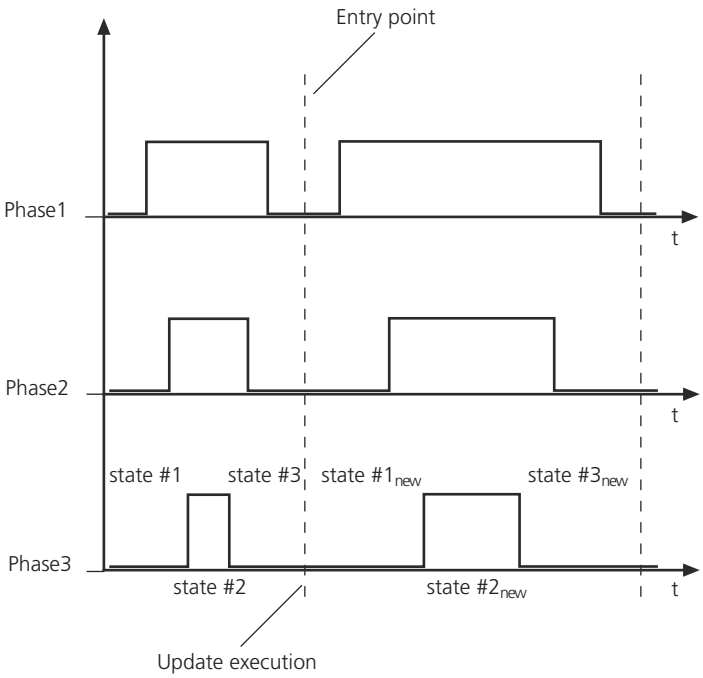
Implementing a loop on page 42), the block mode allows you to perform the update always at the beginning of the loop.

The illustration below shows the block update of `delay` values. New values become effective beginning with the first state of the period.




**Synchronous mode** With the `DS4002_CMD_SYNCDATA` and `DS4002_CMD_SYNCUSE` commands, you can implement the synchronous mode, which is similar to the block mode. Like the block mode, this allows you to update the states of several channels synchronously. The `DS4002_CMD_SYNCDATA` command prepares the update of several channels. Use the `DS4002_CMD_SYNCUSE` command to make the updates effective.

The illustration below shows the synchronous update of the `delay` values of a 3-phase PWM signal. The update is performed synchronously for all the three PWM phases.



For details on programming the synchronous mode, refer to ds4002_update_state (DS4002 RTLib Reference 🕮).

**Note**

Before you execute the **DS4002_EXEC_CMD** macro, always update *all* parameters that can change even if they remain constant during the update. This applies to all update modes. For example, do not use the following construction:

```
ds4002_update_state(…,channel1,state1,…);
DS4002_EXEC_CMD(…,DS4002_CMD_IMMEDIATE,channel1);

ds4002_update_state(…,channel1,state2,…);
DS4002_EXEC_CMD(…,DS4002_CMD_IMMEDIATE,channel1);
```

First, **state1** is updated correctly. Then **state2** is updated, but – due to the swinging buffer principle used by the DS4002 – **state1** simultaneously takes on the parameter values before the first update. To avoid this, use the following construction:

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

For details on the swinging buffer principle, refer to Swinging Buffer Principle on page 84.

**Related topics**

Basics

Basics of the Timing I/O Unit....20
Defining and Specifying States....39
Generation of Arbitrary Signals....36
Starting Signal Generation....45

Examples

Example of Generating an Arbitrary Signal....49

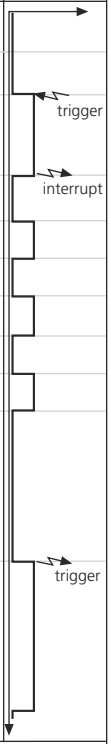
References

DS4002_EXEC_CMD (DS4002 RTLib Reference 🕮)
ds4002_update_state (DS4002 RTLib Reference 🕮)

# Example of Generating an Arbitrary Signal

**Example**

As an example of programming an arbitrary pulse pattern, the following table lists the states of a state machine code, the program flow and the resulting digital signal:

| State | Delay[1)] | Level[1)] | Trigger[1)] | Instr[1)] | Count[1)] | Addr[2)] | Loop Counter | Signal |
|---|---|---|---|---|---|---|---|---|
| 0 | DS4002_DELAY(1e-3) | DS4002_LOW | 0 | DS4002_CONTINUE | 0 | 0 | |  |
| 1 | DS4002_WAIT | DS4002_HIGH | 0 | DS4002_CONTINUE | 0 | 1 | | |
| 2 | DS4002_DELAY(2e-3) | DS4002_LOW | DS4002_INTERRUPT | DS4002_LOADCOUNTER | 3 | 2 | 3 | |
| 3 | DS4002_DELAY(1e-3) | DS4002_HIGH | 0 | DS4002_CONTINUE | 0 | 4 | 3 | |
| 4 | DS4002_DELAY(1e-3) | DS4002_LOW | 0 | DS4002_REPEAT | 0 | 5 | 3 | |
| 3 | | | | | | | 2 | |
| 4 | | | | | | | 2 | |
| 3 | | | | | | | 1 | |
| 4 | | | | | | | 1 | |
| 5 | DS4002_DELAY(4e-3) | DS4002_HIGH | DS4002_MASK(2) | DS4002_CONTINUE | 0 | 6 | 0 | |
| 6 | DS4002_DELAY(4e-3) | DS4002_LOW | 0 | DS4002_JUMP | 1 | 7 | | |



State 0: After 1 ms, set output to low level.
State 1: Wait for trigger event. After the trigger event, set output to high level.
State 2: After 2 ms, generate interrupt (only possible for channel 1 and 2), set output to low level and load the counter with 3.
State 3: After 1 ms, set output to high level. This state is located at address 4, since state 2 requires 2 words in the state machine code.
State 4: After 1 ms, set output to low level. Decrement counter by 1. If counter value > 0, proceed to the state defined after DS4002_LOADCOUNTER command. Otherwise proceed to the next state.
State 5: After 4 ms, set output to high level and trigger channel 2.
State 6: After 4 ms, set output to low level and jump to the state with address 1. This jump instruction closes the infinite loop

1) Parameter of ds4002_define_state.

2) Return value of ds4002_define_state.

## Related topics

Basics

Basics of the Timing I/O Unit....20
Defining and Specifying States....39
Generation of Arbitrary Signals....36
Updating State Parameters....46

# Triggering

## Triggering the Start of Signal Generation Externally

**Introduction**

The DS4002 provides two external trigger input signals, TRIGA and TRIGB. This allows you to connect two devices externally to the DS4002, and use them to trigger the start of signal generation on one or more channels of the timing I/O unit.

**Note**

- It is not possible to trigger the start of signal measurement and event capture.
- To trigger Monoflop signals externally, you must use RTLib functions. Monoflop signals cannot be triggered externally via RTI blocks.

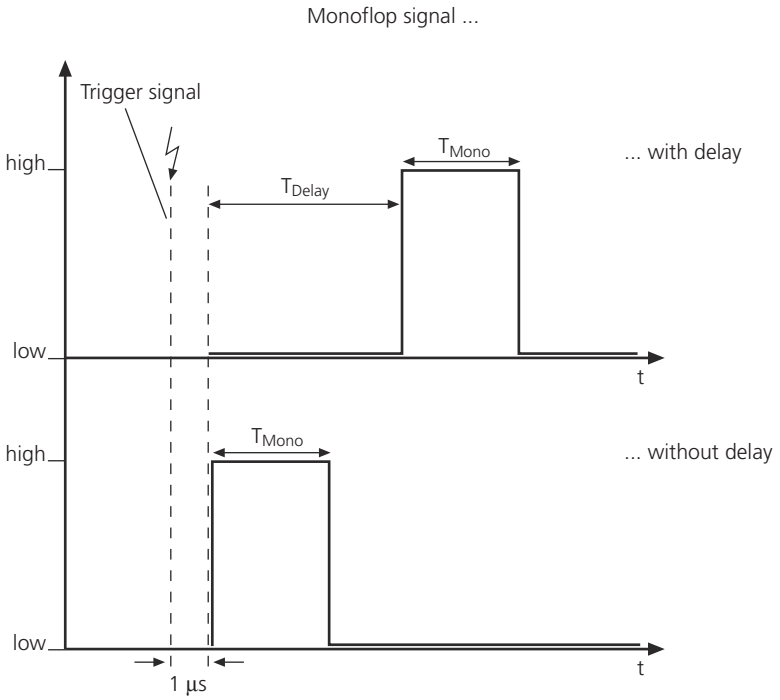
**Enabling and performing triggering**

To use an external trigger signal, you have to enable it with RTLib's `ds4002_ext_trigger_set` function beforehand.

Monoflop signal generation and the generation of arbitrary signals can be triggered externally. For the generation of arbitrary signals, the corresponding state machine code must contain a state with the `DS4002_WAIT` constant as the `delay` parameter. The output signal is not changed until the trigger event.

**Constant delay**

If signal 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. The constant delay results from the time gap between the occurrence of the trigger's edge condition and the detection of the trigger by the DS4002. The following illustration shows an externally triggered monoflop signal as an example.




**Synchronous signal generation**

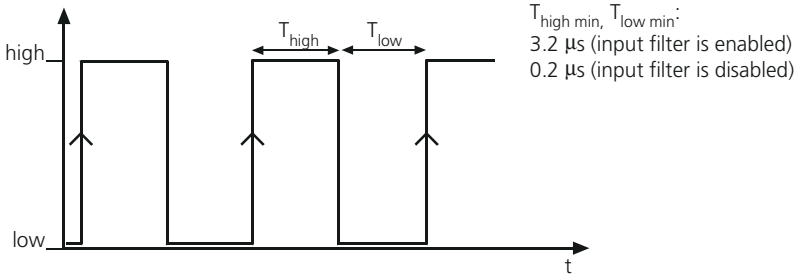
If you use one external trigger event to trigger the start of signal generation on several channels, this is done synchronously. To ensure that the signals remain synchronous, the delay of each state executed after the trigger event should be specified according to the rule of thumb for minimum delay values. This allows the controller to process all the channels of the timing I/O unit.

For information on the rule of thumb and the effects of specifying too small delay values, refer to Limitations Due to the Controller Processing Time on page 91.

**Characteristics of the external trigger signal**

The external trigger signal must have TTL input voltage level. Signal generation is triggered on the rising edge of the external signal. Because the inputs are sampled at 5 MHz, the trigger pulse must be high for at least 0.2 µs to be recognized. This also applies to the interval before the next trigger pulse.

External trigger signals are affected by optional input signal filtering. With input signal filtering, the trigger duration and the interval before the next trigger pulse must be at least 3.2 µs instead of 0.2 µs. The illustration below shows the timing requirements with and without input signal filtering.




For details, refer to Input Signal Filtering on page 64.

**RTI/RTLib support**

You can enable external triggering via RTLib for the DS4002. For details, see External Triggering (DS4002 RTLib Reference 🕮).

Using RTI, you can enable external triggering by incorporating the `ds4002_ext_trigger_set` function in a Simulink S-function. For details, refer to Implementing S-Functions (RTI and RTI-MP Implementation Guide 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTLib functions and the corresponding pins used to provide the external trigger signal.

| Related RTLib Function | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|
| See ds4002_ext_trigger_set (DS4002 RTLib Reference 🕮) | P2 17 | CP9 | TRIGA |
| | P2 50 | CP10 | TRIGB |

**Related topics**

Basics

Basics of Generating Arbitrary Signals..... 37

References

ds4002_ext_trigger_set (DS4002 RTLib Reference 🕮)

# Measurement of Digital Signals and Event Capture

**Introduction**

The timing I/O unit of the DS4002 can be used to measure and capture digital signals and events on up to 8 channels.

**Where to go from here**

Information in this section



Information in other sections

# Overlap and Contiguous Read Modes

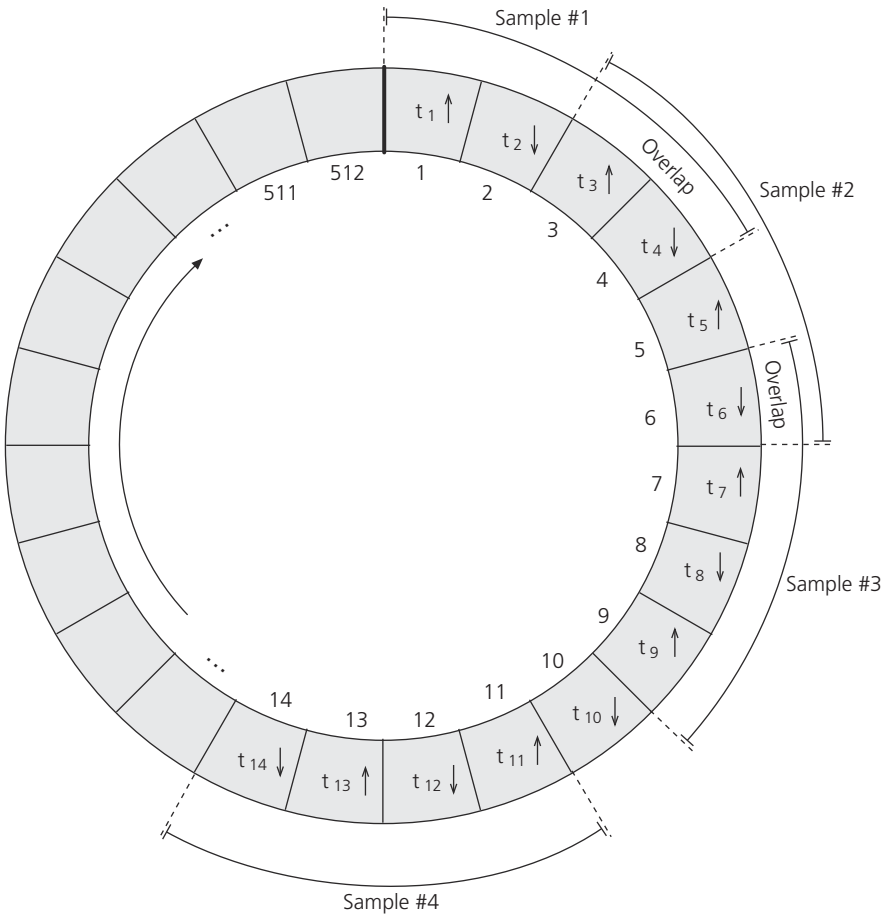
**Introduction**

The DS4002 provides two different modes to read from the event buffer that is configured as a circular buffer: Once it is filled, it always contains the data for the most recent 512 events. Older data is overwritten.

**Overlap mode**

In the *overlap mode*, it is always the same number of events – $count_{Events}$ – that is read from the event buffer. If the number of events captured between two sample steps is lower than $count_{Events}$, there is an overlap, which means that some events are read again.

Reading from the event buffer starts with the event captured last. For sample #2 in the illustration below, $t_6$ is read first, $t_3$ is read last; $count_{Events} = 4$.



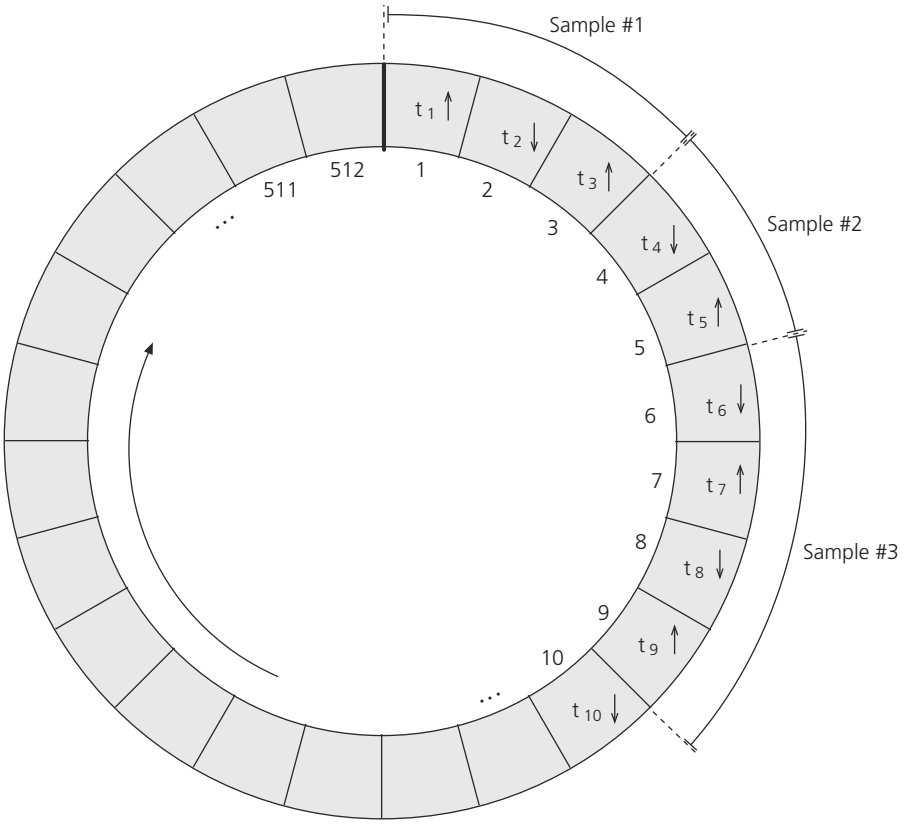

**Note**

If the number of events captured between two sample steps is higher than $count_{Events}$, a part of the captured events gets lost. In the illustration above, this applies to the event $t_{10}$.

**Contiguous mode**

In the *contiguous mode*, only the events available since the last sample step are read from the event buffer. Measurements therefore never overlap. The contiguous mode allows you to capture events continuously. You can specify the maximum number of events $count_{Events}$ that is read from the event buffer.

Reading from the event buffer starts with the first event that was not yet read. For sample #3 in the illustration below, $t_6$ is read first since $t_5$ was already read for sample #2. Reading from the event buffer for sample #3 ends with $t_9$.




**Note**

- If the number of events captured between two sample steps is higher than 512, some of the captured events get lost (buffer overflow). For details, refer to Event Data Capture (DS4002 RTLib Reference 🕮).
- If you use DS4002 Blockset, you can choose the contiguous mode only for the **DS4002READ_EVENT_Bx_Cy** block. The other RTI blocks used for signal measurement (**DS4002FTOD**, **DS4002PWMTOD**) always work in the overlap mode.

**Related topics**

Basics

Basics of the Timing I/O Unit.....20
Measurement of Digital Signals and Event Capture.....54

References

DS4002FTOD (DS4002 RTI Reference 🕮)
DS4002PWMTOD (DS4002 RTI Reference 🕮)
DS4002READ_EVENT_Bx_Cy (DS4002 RTI Reference 🕮)

# PWM Signal Measurement (PWM2D)

**Introduction**

The timing I/O unit of the DS4002 provides inputs for the measurement of the average PWM frequency $f_p$ and duty cycle $d$ on up to 8 channels.

**Minimum PWM frequency**

For each channel you use for PWM signal measurement, you can specify a minimum PWM frequency $f_{p,\,min}$. For input signals with a frequency $f_P < f_{P,\,min}$ the DS4002 outputs a frequency value $f_p$ = 0.0 Hz. To disable this behavior, the minimum PWM frequency value $f_{p,\,min}$ = 0.0 Hz has to be specified.

**Note**

When the input signal is turned off or disconnected during measurement, the frequency of the signal actually is 0.0 Hz. If you specified $f_{p,\,min}$ = 0.0 Hz, however, the DS4002 will output:

- The last frequency value measured before the signal was turned off or disconnected (overlap mode), or
- The `DS4002_EMPTY` return code (contiguous mode).

To avoid this effect, specify $f_{p,\,min}$ > 0.0 Hz.

**Maximum PWM frequency**

The maximum PWM frequency you can measure is determined by the processing time of the board controller. For details, refer to Maximum Frequency for Signal Measurement and Event Capture on page 95.

**Contiguous and overlap modes**

If you perform PWM2D, you can specify the number of periods ($count_{Periods}$) which the DS4002 uses to compute $f_p$ and d. For this purpose, the DS4002 captures both the rising and falling edges of the input signal. You can choose between the overlap and the contiguous mode. The range of $count_{Periods}$ depends on the mode you choose:

| Read Mode | Range for $count_{Periods}$ |
|---|---|
| Overlap | 1 … 250 |
| Contiguous | 1 … 150 |

For more details on the two read modes, see Overlap and Contiguous Read Modes on page 55.

**Note**

If you use RTI's DS4002PWMTOD block, the overlap mode is used automatically. The contiguous mode is not supported by RTI.

**Measuring symmetric PWM signals**

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 on page 96.

**Read event interrupt**

Using RTLib4002, you can specify the number of measured edges after which an interrupt from the DS4002 to the connected processor board is generated. For details, refer to Read Event Interrupt on page 81.

**RTI/RTLib support**

You can perform PWM signal measurement via DS4002 Blockset and RTLib. For details, refer to

- RTI: **DS4002PWMTOD** (DS4002 RTI Reference 🕮)
- RTLib: PWM Signal Measurement (PWM2D) (DS4002 RTLib Reference 🕮)

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI block and the RTLib functions and the corresponding pins used for PWM signal measurement.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002PWMTOD | Ch 1 | See PWM Signal Measurement (PWM2D) (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit.....20

References

DS4002PWMTOD (DS4002 RTI Reference 🕮)

## Square-Wave Signal Measurement (F2D)

**Introduction**

The timing I/O unit of the DS4002 provides inputs for the measurement of the average square-wave signal frequency f on up to 8 channels.

**Minimum F2D frequency**

For each channel you use for square-wave signal measurement, you can specify a minimum frequency $f_{min}$. For input signals with a frequency $f < f_{min}$ the DS4002 outputs a frequency value f = 0.0 Hz. To disable this behavior, the minimum square-wave frequency value $f_{min}$ = 0.0 Hz has to be specified.

**Note**

When the input signal is turned off or disconnected during measurement, the frequency of the signal actually is 0.0 Hz. If you specified $f_{p,\,min}$ = 0.0 Hz, however, the DS4002 will output:

- the last frequency value measured before the signal was turned off or disconnected (overlap mode), or
- the `DS4002_EMPTY` return code (contiguous mode).

To avoid this effect, specify $f_{p,\,min} > 0.0$ Hz.

**Maximum F2D frequency**

The maximum F2D frequency you can measure is determined by the processing time of the board controller. For details, refer to Maximum Frequency for Signal Measurement and Event Capture on page 95.

**Contiguous and overlap modes**

If you perform F2D, you can specify the number of periods – $count_{Periods}$ – which the DS4002 uses to compute f. For this purpose, the DS4002 captures the rising edges of the input signal. You can choose between the overlap and the contiguous mode. The range of $count_{Periods}$ depends on the mode you choose:

| Read mode | Range for $count_{Periods}$ |
|---|---|
| Overlap | 1 … 511 |
| Contiguous | 1 … 500 |

For more details on the two read modes, see Overlap and Contiguous Read Modes on page 55.

**Note**

If you use RTI's **DS4002FTOD** block, the overlap mode is used automatically. The contiguous mode is not supported by RTI.

**Read event interrupt**

Using RTLib4002, you can specify the number of measured edges after which an interrupt from the DS4002 to the connected processor board is generated. For details, refer to Read Event Interrupt on page 81.

**RTI/RTLib support**

You can perform square-wave signal measurement via DS4002 Blockset and RTLib. For details, refer to

- DS4002FTOD (DS4002 RTI Reference 🕮)
- Square-Wave Signal Measurement (F2D) (DS4002 RTLib Reference 🕮)

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI block and the RTLib functions and the corresponding pins used for square-wave signal measurement.

The I/O features of the DS4002 conflict with each other. For details, refer to Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002FTOD | Ch 1 | See Square-Wave Signal Measurement (F2D) (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

### Related topics

Basics

Basics of the Timing I/O Unit....20
Measurement of Digital Signals and Event Capture....54

References

DS4002FTOD (DS4002 RTI Reference 🕮)

## 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.

You specify one channel as the reference signal, and the second as the compare signal. The result is scaled in rad and mapped into the interval + π … - π:

- ΔΦ > 0 indicates that the compare signal leads the reference signal.
- ΔΦ < 0 indicates that the compare signal lags the reference signal.

To get reasonable values for ΔΦ, the frequencies of the reference signal and the compare signal must be equal.

### Number of periods

If you perform phase-shift measurement, you can specify the number of periods – $count_{Periods}$ – which the DS4002 uses to compute ΔΦ, in the range 1 … 509. Phase-shifts are measured in the overlap mode. See Overlap and Contiguous Read Modes on page 55 for details.

**RTI/RTLib support**

You can perform phase-shift measurement via DS4002 Blockset and RTLib. For details, see

- DS4002PHASE_Bx_Cry_Cmz (DS4002 RTI Reference 🕮)
- Phase-Shift Measurement (DS4002 RTLib Reference 🕮)

**Example**

For an example based on RTLib4002, refer to Example of Using the Phase-Shift Measurement Functions (DS4002 RTLib Reference 🕮).

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTLib functions and the corresponding pins used for phase-shift measurement.

The I/O features of the DS4002 conflict with each other. For details, see Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002PHASE_Bx_Cry_Cmz | Ch 1 | See Phase-Shift Measurement (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit ..... 20
Measurement of Digital Signals and Event Capture ..... 54

# Event Capture

**Introduction**

To determine the characteristics of arbitrary digital input signals, you can directly access the event buffer of the DS4002 for further processing. The event buffer holds the direction of captured edges – rising or falling edge – and the corresponding time stamps. For details, refer to Event Buffer on page 86.

**Maximum event capture count rate**

The maximum rate for events to be captured is determined by the processing time of the board controller. For details, refer to Maximum Frequency for Signal Measurement and Event Capture on page 95.

**Contiguous and overlap modes**

If you perform event capture, you can specify the number of events the DS4002 reads – $count_{Events}$ – and choose between the overlap and the contiguous mode. The range of $count_{Events}$ depends on the mode you choose:

| Read mode | Range for $count_{Events}$ |
|---|---|
| Overlap | 1 … 511 |
| Contiguous | 1 … 300 |

For more details on the two read modes, see Overlap and Contiguous Read Modes on page 55.

**Read event interrupt**

Via DS4002 Blockset and RTLib for the DS4002, you can specify the number of measured events after which an interrupt from the DS4002 to the connected processor board is generated. For details, refer to Read Event Interrupt on page 81.

**RTI/RTLib support**

You can perform event capture via DS4002 Blockset and RTLib. Refer to

- DS4002READ_EVENT_Bx_Cy (DS4002 RTI Reference 🕮).
- Event Data Capture (DS4002 RTLib Reference 🕮).

**Execution times**

The execution times required by the RTLib functions have been measured. For details on the results and measurement setup, refer to Function Execution Times (DS4002 RTLib Reference 🕮).

**Connecting external devices**

For a circuit diagram and information on the electrical characteristics and signal conditioning of the timing I/O unit, refer to Signal Connection to External Devices (PHS Bus System Hardware Reference 🕮).

**I/O mapping**

The following table shows the mapping between the RTI block and RTLib functions and the corresponding pins used for event capture.

The I/O features of the DS4002 conflict with each other. Refer to Conflicting I/O Features on page 99.

| Related RTI Blocks | Ch (RTI) | Related RTLib Functions | Ch (RTLib) | Conn. Pin | Pin on CP | Signal |
|---|---|---|---|---|---|---|
| DS4002READ_EVENT_Bx_Cy | Ch 1 | See Event Data Capture (DS4002 RTLib Reference 🕮) | Ch 1 | P2 14 | CP1 | CH1 |
| | Ch 2 | | Ch 2 | P2 47 | CP2 | CH2 |
| | Ch 3 | | Ch 3 | P2 31 | CP3 | CH3 |
| | Ch 4 | | Ch 4 | P2 15 | CP4 | CH4 |
| | Ch 5 | | Ch 5 | P2 48 | CP5 | CH5 |
| | Ch 6 | | Ch 6 | P2 16 | CP6 | CH6 |
| | Ch 7 | | Ch 7 | P2 49 | CP7 | CH7 |
| | Ch 8 | | Ch 8 | P2 33 | CP8 | CH8 |

**Related topics**

Basics

Basics of the Timing I/O Unit.....20
Measurement of Digital Signals and Event Capture.....54

## Input Signal Filtering

**Introduction**

Via RTLib's `ds4002_enable_filter` function, you can enable input signal filtering globally for all the channels of the timing I/O unit and the external trigger inputs. In this case, input signal changes are recognized only if they last for more than 3.2 µs.

**Example**

If input signal filtering is enabled, square-wave input signals will not be recognized if their frequency is higher than 156.3 kHz (= $(2 \cdot 3.2\ \mu s)^{-1}$).

**RTI/RTLib support**

You can enable input signal filtering via RTLib for the DS4002. For details, see Input Signal Filtering (DS4002 RTLib Reference 🕮).

Using RTI, you can enable input signal filtering by incorporating the `ds4002_enable_filter` function in a Simulink S-function. For details, refer to Implementing S-Functions (RTI and RTI-MP Implementation Guide 🕮).

## Related topics

Basics

Measurement of Digital Signals and Event Capture..... 54

References

ds4002_enable_filter (DS4002 RTLib Reference 🕮)

# Angle-Based Mode

**Introduction**

For applications that are angle-based such as the generation of crankshaft sensor signals, you can switch the operating mode of the timing I/O unit. Implementing the angle-based mode depends on the board revision of your DS4002.

**Where to go from here**

Information in this section

Basics on the Angle Base Mode....66

Implementing the Angle-Based Mode (Board Revision DS4002-01 … 03)....68
You can implement the angle-based mode for DS4002 boards with a board revision DS4002-01 … 03).

Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04)....71
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).

Information in other sections

Basics of the Timing I/O Unit....20
Basic information on the controller serving the timing I/O unit of the DS4002.

## Basics on the Angle Base Mode

## Basics on the Angle Base Mode

**Introduction**

For applications that are angle-based such as the generation of crankshaft sensor signals, you can switch the operating mode of the timing I/O unit. Implementing the angle-based mode depends on the board revision of your DS4002.

The timing I/O unit of the DS4002 can be driven in two different operating modes: time-based mode and angle-based mode.

**Note**

The mode setting applies to all channels of the timing I/O unit. You cannot use both modes at the same time. For example, it is not possible to implement an angle-based application on one channel, and perform PWM signal generation on another channel simultaneously.

## Time-based mode

In the time-based mode, the 30-bit time-base counter of the timing I/O unit is incremented by 1 every 200 ns, which is the time base of the board. For this reason, one cycle of the time-base counter is completed after 214.748 s ( = $2^{30}$ · 200 ns), and the counter turns around to zero.

The time-based mode allows you to implement time-based applications such as PWM signal generation, or square-wave signal measurement. The time-based mode is the default operating mode of the DS4002 after board initialization.

## Angle-based mode

In contrast with the time-based mode, the increment added to the time-base counter every 200 ns is not constant in the angle-based mode. Instead, the increment is proportional to the current speed of a rotating shaft:

**360° mode** In 360° mode, the increment is scaled so that one cycle of the 30-bit time-base counter represents one complete revolution of a rotating shaft.

**720° mode** In 720° mode, the increment is scaled so that one cycle of the 30-bit time-base counter represents two complete revolutions of a rotating shaft.

To use the timing I/O unit in angle-based mode, the rotational speed is therefore required as an additional input parameter.

The angle-based mode allows you to implement angle-based applications such as crankshaft sensor signal generation, or injection pulse measurement.

## Related topics

Basics

Angle-Based Mode on a Single DS4002 ..... 72
Angle-Based Mode on Several dSPACE Boards ..... 73
Basics of the Angle-Based Mode for Board Revision DS4002-01 ... 03 ..... 68

# Implementing the Angle-Based Mode (Board Revision DS4002-01 … 03)

**Introduction**

You can implement the angle-based mode for DS4002 boards with a board revision DS4002-01 … 03).

**Where to go from here**

Information in this section



# Basics of the Angle-Based Mode for Board Revision DS4002-01 ... 03

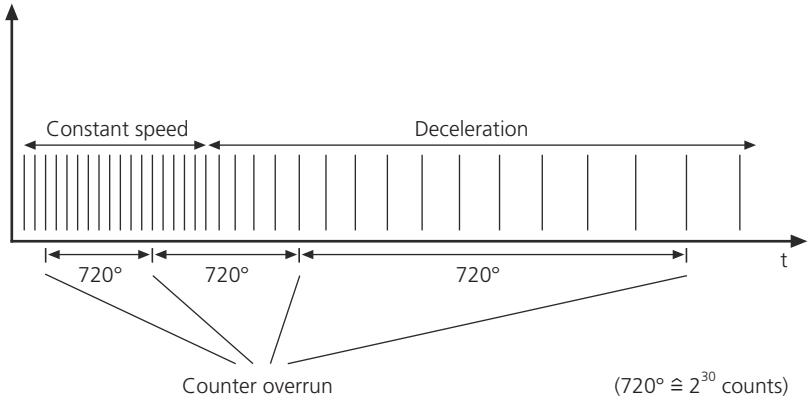
**Introduction**

To work with the DS4002 in angle-based mode (360° mode or 720° mode, see Angle-based mode on page 67), you have to specify the rotational speed.

**Rotational speed input parameter**

To specify the rotational speed, use the `ds4002_set_rpm` (360° mode) or `ds4002_set_rpm2` (720° mode) functions. Using these functions, the increment added to the 30-bit time-base counter of the DS4002 is not constant in the angle-based mode. Instead, it is proportional to the current speed of a rotating shaft. It is scaled so that one cycle of the time-base counter represents one or two complete revolutions of a rotating shaft (360° or 720° mode).

The following drawing illustrates this for the 720° mode:




The functions above therefore require the rotational speed `rpm` as the input parameter. Changes of the rotational speed input parameter come into effect every 200 ns.

**Tip**

Specifying 0.3 rpm (360° mode) or 0.6 rpm (720° mode) as the rotational speed results in a time-base counter increment of 1. This allows you to return to the time-based mode, which is the default mode after board initialization.

**RTI/RTLib support**

You can implement the angle-based mode via RTLib for the DS4002. For details, see Angle-Based Mode (DS4002 RTLib Reference 🕮).

Using RTI, you have to implement the angle-based mode with RTLib4002 functions, and incorporate your C code in a Simulink S-function. For details, refer to Implementing S-Functions (RTI and RTI-MP Implementation Guide 🕮).

**Related topics**

References

ds4002_set_rpm (DS4002 RTLib Reference 🕮)
ds4002_set_rpm2 (DS4002 RTLib Reference 🕮)

## Generating Angle-Based Signals

**Introduction**

To generate angle-based signals, you have to provide the rotational speed. Refer to Rotational speed input parameter on page 68.

**Specifying angle values instead of time values**

After setting the angle-based mode, you can program the desired pulse patterns directly in C code with RTLib4002. For this purpose, you have to implement a state machine code representing the desired pulse pattern. When defining and updating the states of this structure, you have to specify the `delay` parameter. In angle-based mode, you have to specify an angle value instead of a time value. For this purpose, RTLib4002 provides the `DS4002_ANGLE` and the `DS4002_ANGLE2` macros (360° and 720° mode).

Apart from specifying an angle value instead of a delay, angle-based and flexible, time-based signal generation do not differ. Refer to Generation of Arbitrary Signals on page 36.

**Example**

For an example of generating a crankshaft signal, refer to Example of Using Angle-Based Functions (DS4002 RTLib Reference 🕮).

**Related topics**

Basics



References

DS4002_ANGLE (DS4002 RTLib Reference 🕮)
DS4002_ANGLE2 (DS4002 RTLib Reference 🕮)

## Measuring Angle-Based Signals

**Introduction**

To measure angle-based signals, you have to provide the rotational speed. Refer to Rotational speed input parameter on page 68.

**Converting time stamps into angle values**

After setting the angle-based mode, you can implement angle-based signal measurement via event capturing. When events are read, the corresponding time stamps are returned. RTLib4002 provides the `DS4002_TIME2ANGLE` and the `DS4002_TIME2ANGLE2` macros (360° and 720° mode) to convert these time stamps into angle values.

Apart from returning angle values instead of time stamps, angle-based signal measurement and event capturing do not differ. Refer to Event Capture on page 63.

**Example**

For an example of measuring ignition and injection signals, refer to Example of Using Angle-Based Functions (DS4002 RTLib Reference 🕮).

**Related topics**

Basics

Angle-Based Mode....66
Generating Angle-Based Signals....69
Implementing the Angle-Based Mode (Board Revision DS4002-01 … 03)....68

References

DS4002_TIME2ANGLE (DS4002 RTLib Reference 🕮)
DS4002_TIME2ANGLE2 (DS4002 RTLib Reference 🕮)

## Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04)

**Introduction**

Other I/O boards can be synchronized with the board.

DS4002 boards of revision DS4002-04 and higher allow you to operate in angle-based mode – 720° mode only synchronously on one or more DS4002 boards.

**Where to go from here**

Information in this section



# Angle-Based Mode on a Single DS4002

**Introduction**

You can implement the angle-based mode on a single DS4002 without distributing its time base to other dSPACE boards.

**Implementing the angle based mode**

To implement the angle-based mode on a single DS4002 without distributing its time base to other dSPACE boards, do the following:

1. Make sure that the time-base connector of the DS4002 is not connected to other I/O boards.
2. Configure the DS4002 on which you want to implement the angle-based mode as the time-base master. Refer to Configuring Time-Base Master and Time-Base Slaves on page 75.
3. Specify the rotational speed. Refer to Specifying the Rotational Speed on page 75.

**Tip**

You can also implement the angle-based mode on a single DS4002 (board revision DS4002-04 and higher) as described in Implementing the Angle-Based Mode (Board Revision DS4002-01 … 03) on page 68. This allows you to select between the 360° mode and the 720° mode.

**RTI/RTLib support**

You can implement the angle-based mode and time-base distribution via RTLib for the DS4002. For details, see Time Base Distribution (DS4002 RTLib Reference 🕮).

Using RTI, you have to implement the angle-based mode and time-base distribution with RTLib4002 functions, and incorporate your C code in a Simulink S-function. For details, refer to Implementing S-Functions (RTI and RTI-MP Implementation Guide 🕮).

**Related topics**

Basics



# Angle-Based Mode on Several dSPACE Boards

**Introduction**

You can implement the angle-based mode and time-base distribution on several I/O boards.

**Usable I/O boards**

You can connect the time-base bus (engine position bus) of the following I/O boards (the engine position bus of the DS2210 and DS2211 boards is the same as the time-base bus of the DS2302, DS4002, DS5001, and DS5203 boards):

- DS2210
- DS2211
- DS2302 (as of board revision DS2302-04)
- DS4002 (as of board revision DS4002-04)
- DS5001 (as of board revision DS5001-06)
- DS5203 (as of board revision DS5203-05)

**Note**

The DS2302 board cannot be used as the bus master.

**Implementing the angle-based mode**

DS4002 boards of revision DS4002-04 and higher provide a time-base connector. Via the connector, you can set up a network and distribute the time base of one DS4002 to other I/O boards (DS2210, DS2211, DS2302, DS4002, DS5001,

DS5203). This allows you to implement the angle-based mode on several dSPACE boards. Do the following:

1. Connect the time-base connectors to set up the network. Refer to Connecting dSPACE Boards for Time-Base Distribution on page 74.
2. Configure one dSPACE board in the network as the time-base master, all the others as time-base slaves. Refer to Configuring Time-Base Master and Time-Base Slaves on page 75.
3. Specify the rotational speed on the time-base master. Refer to Specifying the Rotational Speed on page 75.

**RTI/RTLib support**

You can implement the angle-based mode and time-base distribution via RTLib for the DS4002. Refer to Time Base Distribution (DS4002 RTLib Reference 🕮).

Using RTI, you have to implement the angle-based mode and time-base distribution with RTLib4002 functions, and incorporate your C code in a Simulink S-function. Refer to Implementing S-Functions (RTI and RTI-MP Implementation Guide 🕮).

**Related topics**

Basics



## Connecting dSPACE Boards for Time-Base Distribution

**Introduction**

To setup the network, you have to connect the appropriate I/O boards physically. For this purpose, these boards are equipped with a time-base connector. Use a standard 26-pin, ribbon cable to setup the network. You can connect the I/O boards to one or several I/O boards.

**Location of the time-base connector**

For the location of the time-base connector on an I/O board, refer to the corresponding *Board Overview* chapter in the PHS Bus System Hardware Reference 🕮.

**Related topics**

Basics

# Configuring Time-Base Master and Time-Base Slaves

**Introduction**

The *time-base master* distributes the time base to other boards – the *time-base slaves*.

**Time-base master, time-base slaves**

You have to configure one of the connected DS4002, DS2210, DS2211 or DS5001 boards as the *time-base master*. You have to configure all the other connected boards as time-base slaves. For this purpose, use the

- `ds4002_apu_mode_set` function for DS4002 boards
- `ds2210_mode_set` function for DS2210 boards
- `ds2211_mode_set` function for DS2211 boards
- `ds5001_apu_mode_set` function for DS5001 boards.

All the other functions including initialization are not affected by the master/slave setting and must be performed for each board individually.

**Note**

- If you want to implement the angle-based mode on a single DS4002 without distributing its time base to other dSPACE boards, you have to configure your DS4002 as the time-base master.
- Do not configure a DS4002 as the time-base master if the network also contains DS2210 or DS2211 boards. Otherwise, the DS2210 or DS2211 board(s) will not work correctly.

**Related topics**

Basics

Angle-Based Mode....66
Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04)....71

References

ds2210_mode_set (DS2210 RTLib Reference 🕮)
ds2211_mode_set (DS2211 RTLib Reference 🕮)
ds4002_apu_mode_set (DS4002 RTLib Reference 🕮)
ds5001_apu_mode_set (DS5001 RTLib Reference 🕮)

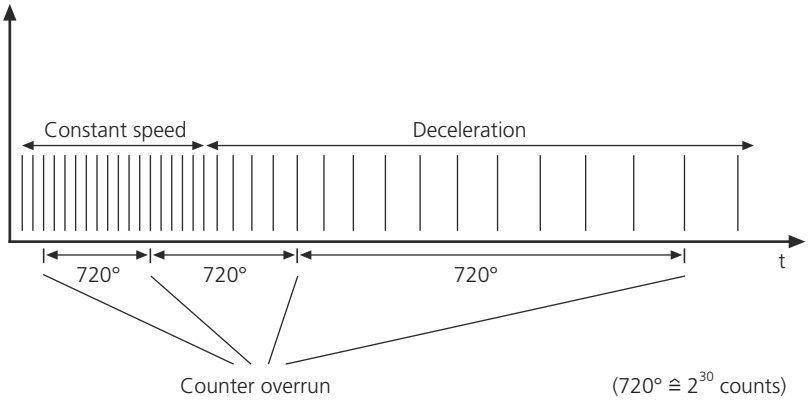
# Specifying the Rotational Speed

**Introduction**

To work with the DS4002 in angle-based mode, you have to specify the rotational speed on the time-base master.

**Rotational speed input parameter**

You can set the `vel` input parameter according to the current speed of a rotating shaft. If a DS4002 is the time-base master, the increment added to the 30-bit time-base counter of the DS4002 is scaled so that one cycle of the time-base counter represents two complete revolutions of the rotating shaft. Run-time changes of the `vel` input parameter come into effect every 200 ns.




The following functions allow you to specify the `vel` input parameter:

- `ds4002_apu_velocity_write` function (time-base master is a DS4002)
- `ds2210_apu_velocity_write` function (time-base master is a DS2210)
- `ds2211_apu_velocity_write` function (time-base master is a DS2211)
- `ds5001_apu_velocity_write` function (time-base master is a DS5001)

**Note**

The rotational speed input parameter used by the `ds4002_apu_velocity_write` function and the `ds4002_set_rpm2` function have the same physical meaning. However, the `vel` parameter used by `ds4002_apu_velocity_write` must be stated in rad/s; the `rpm` parameter used by `ds4002_set_rpm2` must be stated in rpm. Use the following conversion formula:

$vel = 2\pi \cdot rpm/60$

**Tip**

If a DS4002 is the time-base master, specifying 0.06 [rad/s] as the `vel` input parameter results in a time-base counter increment of 1. This allows you to return to the time-based mode, which is the default mode after board initialization.

**Resolution**

According to one cycle of a four-stroke engine, the engine cycle is 0 … 720° (0 … 4π). Although the time-base master internally uses a 30-bit time-base counter, only the most significant 13 bits are supplied to the time-base slaves. The resolution therefore is:

| | Resolution (in bit) | Resolution (in °) |
|---|---|---|
| Time-base master | $2^{30}$ | $6.7 \cdot 10^{-7}$ |
| Time-base slave | $2^{13}$ | 0.088 |

## Related topics

Basics

Angle-Based Mode.....66
Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04).....71

References

ds2210_apu_velocity_write (DS2210 RTLib Reference 🕮)
ds2211_apu_velocity_write (DS2211 RTLib Reference 🕮)
ds4002_apu_velocity_write (DS4002 RTLib Reference 🕮)
ds5001_apu_velocity_write (DS5001 RTLib Reference 🕮)

# Interrupts Provided by the DS4002

**Where to go from here**

Information in this section

Basics on Interrupts of the DS4002 ..... 79
Provides basic information on the interrupts.

Signal Generation Interrupt ..... 80
You have access to two interrupts that can be generated if you perform signal generation on channels 1 or 2 of the timing I/O unit. Depending on the kind of signal you generate you can specify when the interrupt is generated.

Read Event Interrupt ..... 81
You can perform signal measurement and event capturing. For each channel, you can let an interrupt be generated after a predefined number of events were captured.

Information in other sections

Introduction to the Features of the DS4002 ..... 9
Provides an overview of the architecture, features and interface of the DS4002.

## Basics on Interrupts of the DS4002

**Introduction**

The DS4002 provides access to various hardware interrupts – originating either from on-board devices, or from external devices connected to the DS4002.

**Interrupts**

The following interrupts are available:

| Interrupt Type | Description | Refer to |
|---|---|---|
| Signal generation | Interrupt during signal generation (channels 1 and 2 only) | Signal Generation Interrupt on page 80 |
| Read event | Interrupt when a predefined number of edges was captured | Read Event Interrupt on page 81 |

**Interrupt processing**

Via the interrupt lines of the PHS bus, interrupts from the DS4002 are sent to the interrupt controller of the connected processor board. Using RTI, the interrupts of the DS4002 can therefore be used to implement interrupt-driven tasks.

**Related topics**

Basics

Read Event Interrupt....81
Signal Generation Interrupt....80
Tasks Driven by Interrupt Blocks (RTI and RTI-MP Implementation Guide 🕮)

# Signal Generation Interrupt

**Introduction**

You have access to two interrupts that can be generated if you perform signal generation on channels 1 or 2 of the timing I/O unit. Depending on the kind of signal you generate you can specify when the interrupt is generated.

| Signal | Interrupt Generation |
|---|---|
| 1-phase PWM signal | On the rising or falling edge of the high-active PWM signal. |
| 3-phase PWM signal | In the middle of the high or low period of the corresponding PWM3 signal. |
| Square-wave signal | On the rising or falling edge of the high-active D2F signal. |
| Monoflop signal | On the rising or falling edge of the high-active monoflop signal. |
| Arbitrary signal | At the end of a state. |

**RTI/RTLib support**

You can access the signal generation interrupts via RTI and RTLib.

- RTI: For details, see DS4002_HWINT_Bx_Iy (DS4002 RTI Reference 🕮)

Depending on the kind of signal you generate, your Simulink model must also contain one of the following blocks to implement interrupt generation via RTI:

- DS4002PWM1_OUT
- DS4002PWM3_OUT
- DS4002MONO_Bx_Cy
- DS4002DTOF

**Note**

Generation of arbitrary signals is only supported by RTLib4002. However, you can program signal generation and interrupt generation in C code, and incorporate the code in a Simulink S-function. In this case, you can access the corresponding interrupt in a Simulink model using the DS4002_HWINT_Bx_Iy block.

- RTLib: Depending on the kind of signal you generate, see `ds4002_pwm_int_init`, `ds4002_pwm3_int_init`, `ds4002_d2f_int_init`, `ds4002_delayed_mono_int_init`, or `ds4002_define_state` in the *DS4002 RTLib Reference*.

**Related topics**

Basics



# Read Event Interrupt

**Introduction**

You can perform signal measurement and event capturing on up to 8 channels. For each channel, you can let an interrupt be generated after a predefined number of events were captured.

**Note**

Using DS4002 Blockset, only event capturing via the DS4002READ_EVENT_Bx_Cy block supports interrupt generation. Interrupt generation for PWM2D or F2D is possible via RTLib for the DS4002 only, not via DS4002 Blockset.

**RTI/RTLib support**

You can access the read event interrupt via RTI and RTLib.

- RTI: For details, refer to DS4002_HWINT_Bx_Iy (DS4002 RTI Reference 🕮)

  To implement read event interrupt generation via DS4002 Blockset, the DS4002READ_EVENT_Bx_Cy block must also be part of your Simulink model.

- RTLib: Depending on the kind of signal you measure, see ds4002_pwm2d_init, ds4002_f2d_init, or ds4002_read_init (DS4002 RTLib Reference 🕮).

**Note**

Up to 8 channels can provide read event interrupts. However, the PHS bus connecting the DS4002 to the processor board provides only one physical interrupt line. If you use RTLib4002, you have to use subinterrupt handling functions of the dSPACE processor board to identify the originating channel of the interrupts. Refer to Subinterrupt Handling (DS1006 RTLib Reference 🕮) or Subinterrupt Handling (DS1007 RTLib Reference 🕮).

## Related topics

Basics

Interrupts Provided by the DS4002 ..... 79
Signal Generation Interrupt ..... 80

References

DS4002READ_EVENT_Bx_Cy (DS4002 RTI Reference 🕮)

# Memory Features

**Introduction**

The DS4002 is equipped with a dual-port memory that is used for signal generation and measurement.

**Where to go from here**

Information in this section



## Memory Usage

**Introduction**

The DS4002 is equipped with a dual-port memory that is used for signal generation and measurement.

**Memory usage for signal generation**

When you perform signal generation, the dual-port memory holds the state machine code with all the parameters and settings for interrupt generation and

internal triggering. To update these parameters during run time, the dual-port memory internally uses the swinging buffer principle.

**Memory usage for signal measurement**

When you perform signal measurement and event capture, the dual-port memory provides each channel of the board's timing I/O unit with an *event buffer*.

**Related topics**

Basics

Basics of Generating Arbitrary Signals..... 37

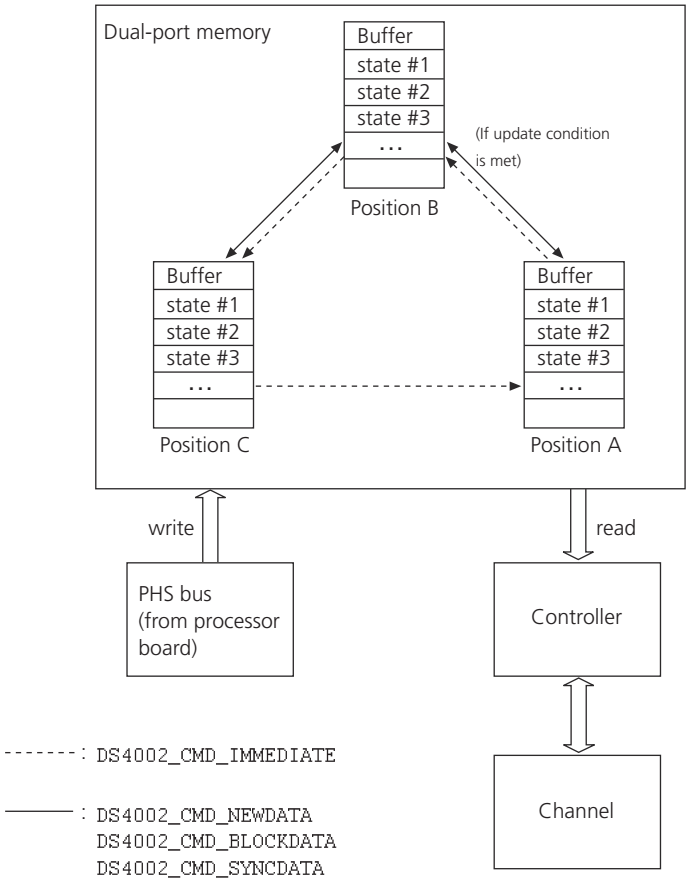
## Swinging Buffer Principle

**Introduction**

For parameter updates during run time, the dual-port memory internally uses the swinging buffer principle for each channel of the board's timing I/O unit. This avoids inconsistencies during parameter updates.

**Reading from and writing to the memory**

The illustration below shows the structure of the three buffers for one channel. New parameter values coming from the processor board via PHS bus are always written to the buffer at position C. The controller always reads the buffer at position A.




**Initial parameter values**

According to the programming structure (refer to Programming structure on page 37) for signal generation, you define the states of a state machine code first using the ds4002_define_state function. Then you load the state machine code to the desired channel of the DS4002 with the ds4002_load_states function. The initial parameter values of the states are loaded to all the three buffers, and the controller reads the buffer at position A.

**Parameter value update**

Parameter values can be updated with the `ds4002_update_state` function. This writes the new values to the buffer at position C.

During parameter update, the DS4002 exchanges only the buffer positions so that no buffer needs to be copied from one position to another. The way in which buffer positions are exchanged depends on the update mode you specify with the `DS4002_EXEC_CMD` macro:

| Update Mode | Update Command | Position C is Changed to … | Position A is Changed to … | Position B is Changed to … |
|---|---|---|---|---|
| Immediate update | `DS4002_CMD_IMMEDIATE` | Position A | Position B | Position C |
| Next delay update<br>Block update<br>Synchronous update | `DS4002_CMD_NEWDATA`<br>`DS4002_CMD_BLOCKDATA`<br>`DS4002_CMD_SYNCDATA` | Position B | Position A (position remains unchanged until update condition is met) | Position C |

A parameter update is complete when the new values are at position A.

- In the immediate mode, the controller uses new values immediately since buffer position C is directly changed to position A.
- In the other modes, the position of the buffer containing new values is first changed from C to B. The position is changed to A when the update condition of the corresponding update mode is met. In the block mode, for example, the position is not changed to A until the next `DS4002_GOTO` command. For details, refer to Updating State Parameters on page 46.

**Related topics**

Basics

Basics of Generating Arbitrary Signals....37
Updating State Parameters....46

References

ds4002_define_state (DS4002 RTLib Reference 🕮)
DS4002_EXEC_CMD (DS4002 RTLib Reference 🕮)
ds4002_load_states (DS4002 RTLib Reference 🕮)
ds4002_update_state (DS4002 RTLib Reference 🕮)

# Event Buffer

**Introduction**

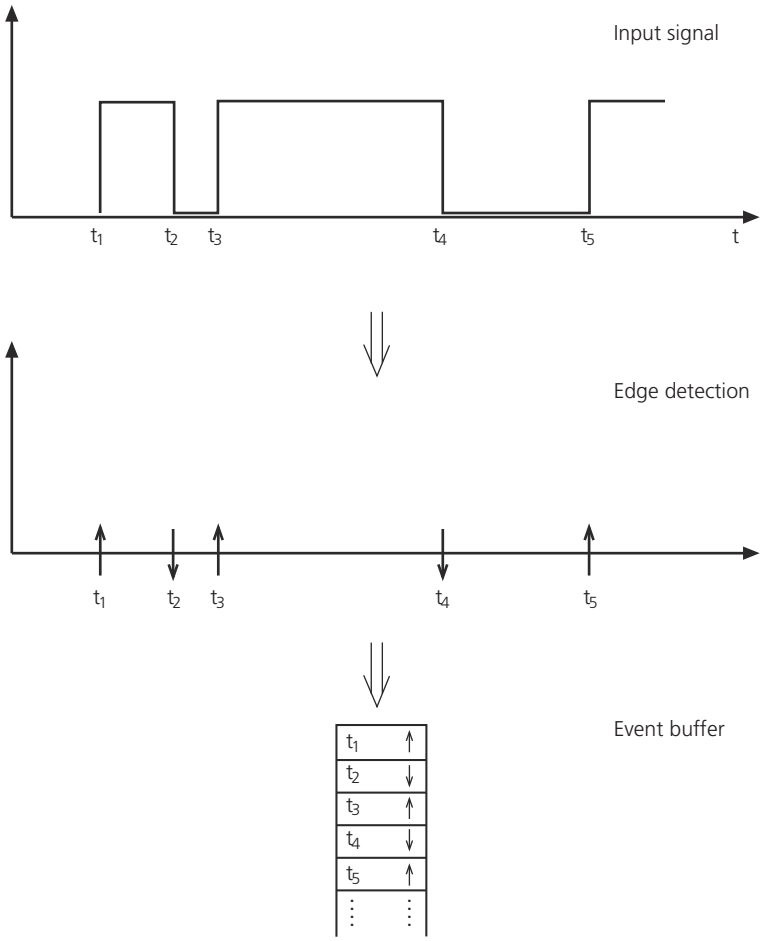
Each digital signal can be described as a series of rising and falling edges. A rising or falling edge together with the corresponding time stamp represent an event. To store the shape of an input signal via the DS4002, the *event buffer* provided by the board's dual-port memory is used.

**Event buffer**

Each digital signal can be described as a series of rising and falling edges. A rising or falling edge together with the corresponding time stamp represent an event. For signal measurement and event capture, the DS4002 provides an event buffer that is configured as a circular buffer: Once the buffer is filled, it always

contains the data for the last 512 events. Old data is overwritten. For each event, the buffer stores the direction of the edge (rising or falling), and the corresponding time stamp (30-bit width) with 200 ns resolution.

The illustration below shows how a digital input signal is stored.




**Read modes**

There are two different modes to read the event buffer: see Overlap and Contiguous Read Modes on page 55.

**Related topics**

Basics

Measurement of Digital Signals and Event Capture.....54

# Limitations

**Introduction**

There are some limitations when you work with the DS4002.

**Where to go from here**

Information in this section

# DS4002 Board Revision

**Introduction**

There are limitations depending on the board revision of the DS4002.

**Time-base connector**

All features related to the time-base connector, which is part of the timing I/O unit, are supported only for board revision DS4002-04 and higher. For details on the affected features, refer to Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04) on page 71.

**Related topics**

Basics

Connecting dSPACE Boards for Time-Base Distribution ..... 74
Implementing the Angle-Based Mode and Time-Base Distribution (Board Revision as of DS4002-04) ..... 71

# Quantization Effects

**Introduction**

Signal generation and measurement are only feasible within the limits of the 200 ns time base of the timing I/O unit, which causes quantization errors that increase with increasing frequencies.

When performing square-wave signal generation (D2F), for example, you will encounter considerable deviations between the desired frequency $f_{desired}$ and the generated frequency $f_{generated}$, especially for higher frequencies. The (quantized) generated signal frequency can be calculated according to the following equation:

$f_{generated} = 1 / (n \cdot R)$

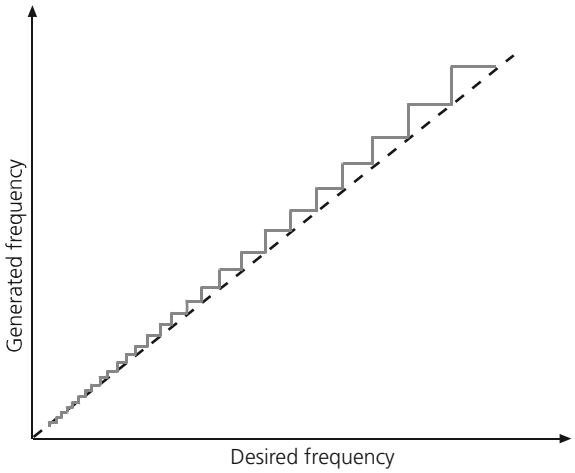
where R is the 200 ns time base, and n is the integer part of $(1 / (f_{desired} \cdot R))$

**Example**

Suppose you want to generate a square-wave signal with $f_{desired}$ = 300 kHz. Calculating the integer part of (1/(300 kHz · 200 ns)) yields n = 16. According to $f_{generated} = 1/(n \cdot R)$, the generated frequency is 312.5 kHz.

**Quatization effect**

The following illustration shows the increasing quantization effect for increasing desired frequencies:




## Limitations Due to the Controller Processing Time

**Introduction**

Proper signal generation and measurement strongly depends on the number of used channels and on the maximum frequency of the signals you want to generate and measure. This is due to the request processing time of the controller and the priority scheme.

**Channel requests**

The channels of the timing I/O unit can generate or measure digital signals (output or input mode). Channels send a request to the controller when a state is completely executed, that is, when the specified delay has expired (output mode), or when a rising or falling edge was captured (input mode). Refer to Channel requests on page 21.

**Command requests**

The controller serves also the command requests for run-time parameter updates from the processor board (output mode). There are two groups of commands:

| Group 1 Commands | Group 2 Commands |
|---|---|
| DS4002_CMD_IMMEDIATE | DS4002_CMD_SYNCDATA |
| DS4002_CMD_NEWDATA | DS4002_CMD_SYNCUSE |
| DS4002_CMD_BLOCKDATA | |

The controller serves group 1 command requests with a higher priority than group 2 command requests. See below for details.

### Request processing time

To process requests for a channel or command, the controller of the DS4002 requires processing times.

**Channel request** The following table shows the time for a channel request.

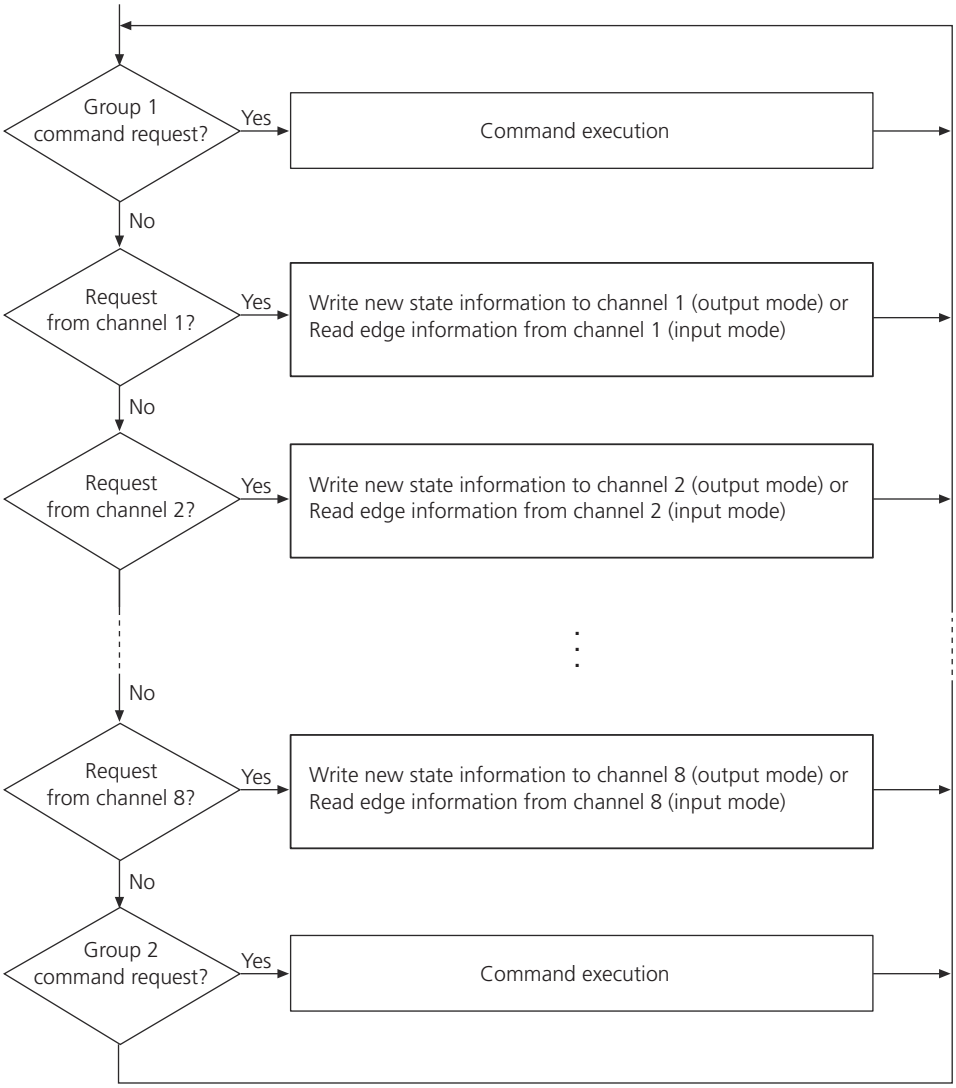
| Channel Request | Time |
|---|---|
| Channel in output mode | 400 ns |
| Channel in input mode | 350 ns |

**Command request** The following table shows the time for a command request.

| Command Request | Time |
|---|---|
| `DS4002_CMD_IMMEDIATE` | 300 ns |
| `DS4002_CMD_NEWDATA` | 100 ns |
| `DS4002_CMD_BLOCKDATA` | 50 ns |
| `DS4002_CMD_SYNCDATA` | 100 ns |
| `DS4002_CMD_SYNCUSE` | 100 ns |

**Priority-based processing of requests**

The controller uses a priority scheme to process channel requests and command requests. The scheme is shown in the flow chart below:




For the effects of the request processing time of the controller and the priority scheme on signal generation and measurement, see

- Minimum Delay Value for Signal Generation on page 94
- Maximum Frequency for Signal Measurement and Event Capture on page 95

# Minimum Delay Value for Signal Generation

**Introduction**

Due to the priority scheme and the request processing time, proper signal generation strongly depends on the number of used channels and on the minimum delay values of states you use in your application.

**Rule of thumb for $t_{delay}$**

As the minimum delay value for one state, use

$t_{delay,min}$ = 600 ns + ((active_channels – 1) · 400 ns)

If you want to update parameter values during run time, you also have to take into account the processing time and the processing priority for the related command requests: see Request processing time on page 92 and Priority-based processing of requests on page 93.

**Note**

Take into account the low processing priority of group 2 command requests.

**Using too small delay values**

If you use delay values smaller than $t_{delay,min}$, the controller cannot process channel requests and command requests sufficiently fast. As a result, the pulse pattern of the output signals differs from the desired pulse pattern.

**Note**

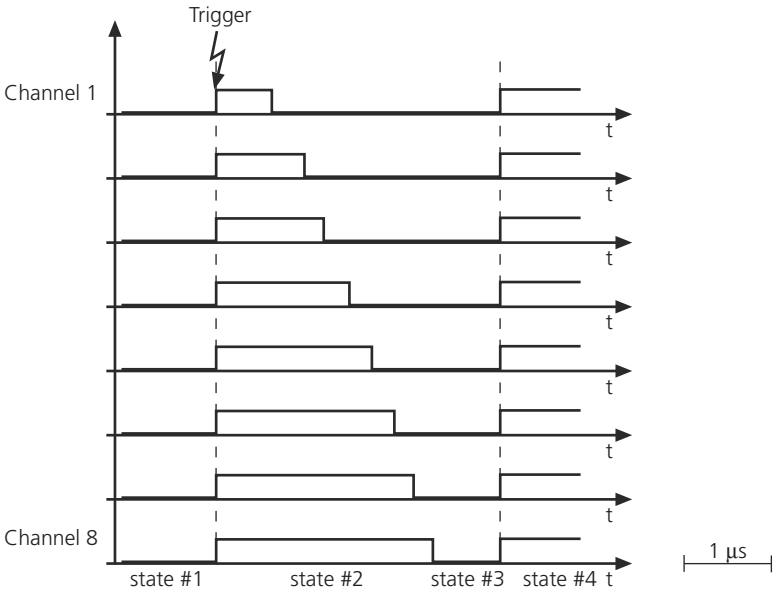
In some cases, signal generation is interrupted for several minutes.

Suppose you program a signal using the following state machine code for all 8 channels:

```
ds4002_define_state(DS4002_WAIT,          DS4002_HIGH,0,DS4002_CONTINUE,0); /* state #1 */
ds4002_define_state(DS4002_DELAY(600e-9), DS4002_LOW, 0,DS4002_CONTINUE,0); /* state #2 */
ds4002_define_state(DS4002_DELAY(4.0e-6), DS4002_HIGH,0,DS4002_CONTINUE,0); /* state #3 */
ds4002_define_state(DS4002_DELAY(5.0e-6), DS4002_LOW, 0,DS4002_GOTO,    0); /* state #4 */
```

After a trigger event, all outputs should immediately be set to high level for 600 ns. After this period, all outputs should be set to low level for 4.0 µs. This period should be followed by a high-level period of 5.0 µs.

Due to the processing time of the controller, the actual pulse pattern of the 8 channels differs significantly from the desired pattern:




After the trigger event, all outputs are immediately set to high level. At the same time, each of the 8 channels sends a request to the controller to get the delay value of state #2.

According to the Priority-based processing of requests on page 93, the controller processes the request of channel 1 first. After 600 ns, the output of channel 1 is set to low level. Then the other channels are processed step by step, and they get the requested information on state #2. The request of channel 8 is processed last after 4 µs. To avoid these effects, use longer delay values according to the rule of thumb (see above).

**Related topics**

Basics

Limitations Due to the Controller Processing Time..... 91

# Maximum Frequency for Signal Measurement and Event Capture

**Introduction**

Due to the priority scheme and the request processing time, proper signal measurement and event capture strongly depend on the number of used channels and on the maximum input frequency (see Priority-based processing of requests on page 93 and Request processing time on page 92).

**Rule of thumb for $f_{max}$**

To estimate the maximum frequency of signals to be measured, use

$f_{max} = (600\ ns + ((active\_channels - 1) \cdot 400\ ns))^{-1}$

**Related topics**

Basics

Limitations Due to the Controller Processing Time..... 91

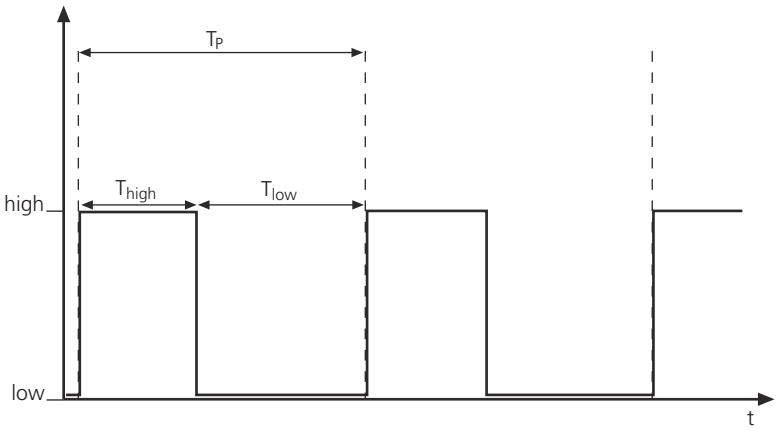
## Limitation for the Measurement of Symmetric PWM Signals

**Introduction**

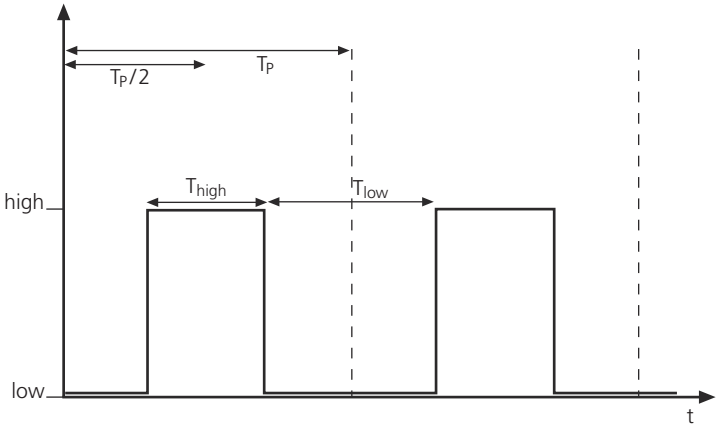
The evaluation of the PWM frequency $f_p$ of symmetric PWM signals is faulty if the duty cycle of the PWM signal changes during measurement.

**Asymmetric and symmetric PWM signals**

PWM measurement is accurate if the PWM period starts with the falling or rising edge of the corresponding PWM signal (asymmetric signal). For example, in the illustration below, each PWM period starts with a rising edge of the asymmetric PWM signal:



The DS4002 can also measure PWM signals that are centered around the middle of the PWM period (symmetric signals):




However, the evaluation of the PWM frequency $f_p$ of symmetric PWM signals is faulty if the duty cycle of the PWM signal changes during measurement.

**PWM frequency evaluation algorithm**

The PWM frequency $f_p$ is evaluated according to the following equation:

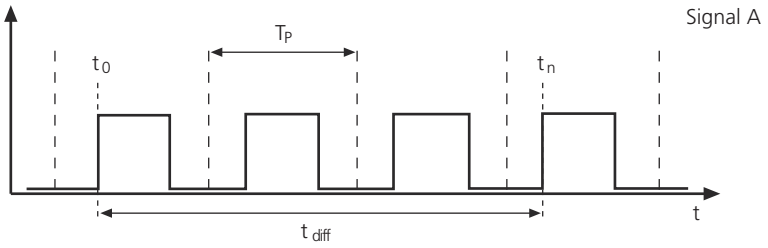
$$f_p = \frac{n}{t_{diff}}$$

Where

$t_{diff}$ is the interval between the first and the last detected rising edge

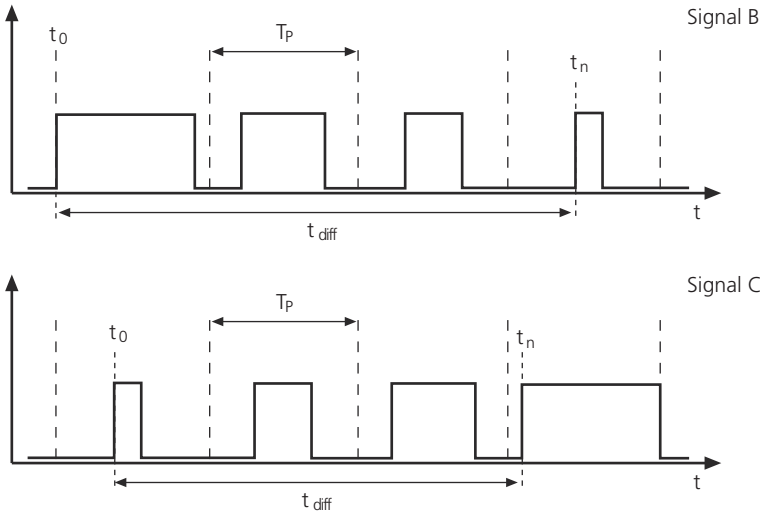
$n$ is the number of PWM periods used to evaluate $f_p$.

The following illustration shows how $f_p$ is evaluated for a symmetric PWM signal (signal A) without duty cycle changes during measurement; $n = 3$:




**Measurement error due to duty cycle changes**

PWM frequency measurement of a symmetric PWM signal is faulty if the duty cycle of the signal changes during measurement. The illustration below shows two PWM signals (signals B and C) with duty cycle changes: The duty cycle of signal B decreases whereas the duty cycle of signal C increases during measurement.




According to the illustration above, duty cycle changes during run time have an effect on the measured interval $t_{diff}$. As a result, the $f_p$ values evaluated for the signals B and C are faulty, since $t_{diff}$ is used to evaluate the PWM frequency $f_p$.

**Estimating the measurement error**

The difference between the correct frequency value and the one evaluated cannot be calculated exactly since it depends on the speed of the duty cycle change. However, the maximum deviation from the correct frequency value $f_p$ can be calculated according to the following equation:

$$f_{deviation} = \pm \frac{f_p}{(2 \cdot \pi - 1)}$$

The evaluated frequency value therefore is in the range

$$f_{p,\ evaluated} = f_p \pm \frac{f_p}{(2 \cdot \pi - 1)}$$

$$f_{p,\ evaluated} = f_p \cdot \left(1 \pm \frac{1}{(2 \cdot \pi - 1)}\right)$$

**Tip**

To decrease the measurement error, specify a large value for $n$, which is the number of PWM periods used to evaluate $f_p$.

# Conflicting I/O Features

**Conflicts for the DS4002**

The following I/O features of the DS4002 use signals CH1 … CH8:

- Signal generation
  - 1-Phase PWM Signal Generation (PWM)
  - 3-Phase PWM Signal Generation (PWM3)
  - Generation of Simple Signals
  - Monoflop Signal Generation
  - Generation of Arbitrary Signals
- Signal measurement and event capture
  - PWM Signal Measurement (PWM2D)
  - Square-Wave Signal Measurement (F2D)
  - Phase-Shift Measurement
  - Event Capture

Each of the signals CH1 … CH8 can be used by only one I/O feature at the same time. For example, if you use CH3 for 1-Phase PWM Signal Generation (PWM), you cannot use this channel to perform PWM Signal Measurement (PWM2D) at the same time.

To perform 3-Phase PWM Signal Generation (PWM3), you have to specify an array of three signals in the range CH1 … CH8. These signals cannot be used for another I/O feature of the DS4002 at the same time.

## Numerics



## A



## B



## C



## D



## E



## F



## G



## H



## I



## L



## M



## N



## O



## P



## R



## S



## T



## U