RapidPro System – I/O Subsystem

MPC565 Implementation Features

For RTI RapidPro Control Unit Blockset

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
  Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/patches for software updates and patches.

Important Notice

Contents

Timing I/O 51

About This Document

Contents

This document provides general information about the system architecture of a RapidPro system containing a RapidPro Control Unit that is used as an I/O subsystem (slave processor MPC565) to complement a rapid control prototyping (RCP) system from dSPACE, such as MicroAutoBox II or a DS1007 modular system (PHS-bus-based system with a DS1007 processor board). It also provides feature-oriented access to the information you need to implement your control models on the dSPACE prototyper using the RapidPro Control Unit.

Required knowledge

It is assumed that you know the features of your rapid control prototyping system, for example, MicroAutoBox II or a PHS-bus-based system.

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/go/help.

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 Rapid Control Prototyping with the RapidPro System

Objective

The RapidPro system is a modular, flexible hardware platform that you can use for different fields of application. One of these is *rapid control prototyping* (RCP), in which the RapidPro Control Unit is used as a slave processor of the existing dSPACE RCP system like MicroAutoBox II or a DS1007 modular system with DS4121 ECU Interface Board.

Where to go from here

Information in this section

Components of a dSPACE RCP System

Objective

Developing a controller involves a lot of optimization effort, which can be reduced by using a rapid control prototyping system (RCP system) like dSPACE Prototyper with the RapidPro system. The RCP hardware replaces the production controller. All sensors and actuators are connected to the RapidPro hardware. You have full authority to control the plant. You can manage the RapidPro hardware with ConfigurationDesk for RapidPro. ControlDesk 3.x or ControlDesk can be used for experimenting with the RCP system. You can add complex I/O interfaces to the Simulink® model of your controller with the RTI and RTLib implementation software. The simulation is done in real time.

Where to go from here

Information in this section

Prototyping Systems with RapidPro Hardware

Objective

The modular, flexible RapidPro hardware platform consists of three units which can also be used separately.

RapidPro Units

A RapidPro system is a combination of the following RapidPro Units:

- RapidPro Control Unit

  Modular microprocessor unit used as an intelligent subsystem to add extra I/O functionality to rapid control prototyping (RCP) hardware, like dSPACE Prototyper (MicroAutoBox II, or DS1007 modular system with DS4121).
- RapidPro SC Unit

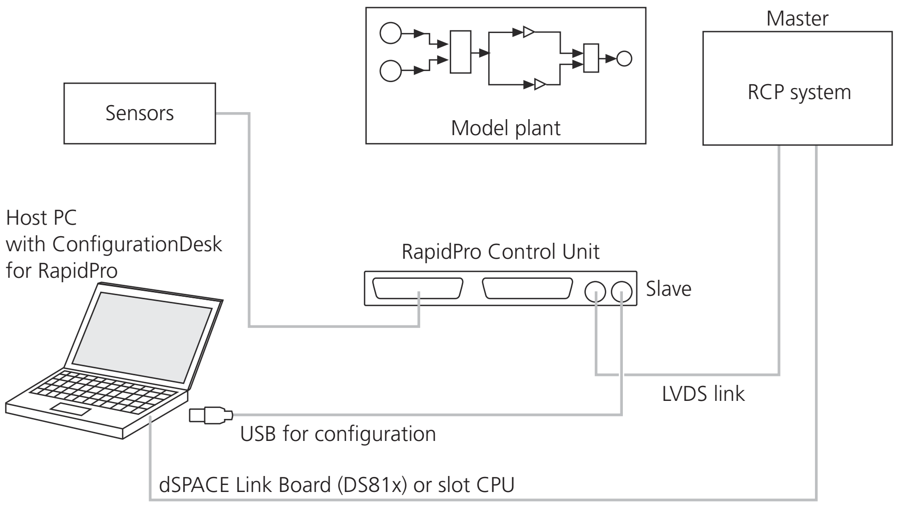
  Modular, intelligent, software-configurable signal conditioning unit
- RapidPro Power Unit

  Modular, intelligent, software-configurable power stage unit

To manage your RapidPro system, you can use ConfigurationDesk for RapidPro.

dSPACE Prototyper with RapidPro Control Unit

The Control Unit with its MPC565 microprocessor can be used to add I/O functionality to your RCP system. The Control Unit has to contain signal conditioning (SC) modules that provide the required I/O functionality. For example, if you want to read digital signals, the Control Unit must contain a signal conditioning module that can handle digital input signals. Some settings of SC modules can be configured, for example, for use with a specific sensor.



Master and slave

Your application is executed on the RCP system (the master processor). The Control Unit serves as a slave processor to realize additional I/O handling independently from the master. The master and the slave are connected by a high-speed Low Voltage Differential Signaling (LVDS) link.

Note

The LVDS link is connected to the ECU interface of the MicroAutoBox II or the DS4121 ECU interface board using a PHS-bus-based system.

- MicroAutoBox II 1401/1504 has no ECU Interface Unit to connect to a RapidPro system.
- A DS4121 provides two ECU Interface Units. Your PHS-bus-based system can contain up to 16 DS4121 boards. It is therefore possible to connect up to 32 RapidPro systems to a PHS-bus-based system.

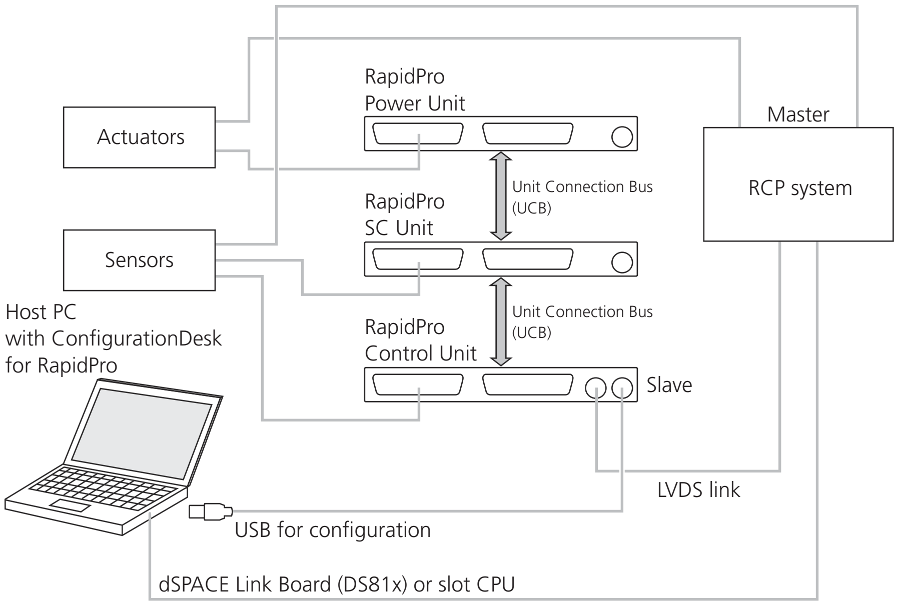
Host PC

The host PC must be connected to the Control Unit and to the master of the RCP system:

- It must be connected to the Control Unit via USB interface when you make the software-configurable settings of the RapidPro hardware using ConfigurationDesk for RapidPro.
- It must be connected to the master when you want to download the application, for example.

dSPACE Prototyper with RapidPro Control Unit, SC Unit and Power Unit

If your application requires more channels for acquiring sensor signals than are provided by the SC modules of the Control Unit, you must add one or more SC Units to your RapidPro system. If your application requires signals to drive actuators and loads, you must add one or more Power Units to your RapidPro system.



Combining units in a stack

Stack without UCB Several single units are mounted mechanically together to build a stack. The stack has no internal electrical connections between the units. Each unit functions as a separate RapidPro system, exactly as when used as a single unit.

Stack with UCB The RapidPro Control Unit is combined with RapidPro SC and/or Power Units in a stack to build a common system. An integrated unit connection bus (UCB) connects the SC and Power Units electrically to the Control Unit without external wiring. The connected units form a RapidPro system stack, which can be managed centrally via the Control Unit. By combining different units and modules, you can build an individual RapidPro system that meets your specific prototyping requirements.

For further information on the RapidPro hardware, refer to the *RapidPro Hardware Installation and Configuration* documents available for the RapidPro system units and modules.

Implementation Software

Objective

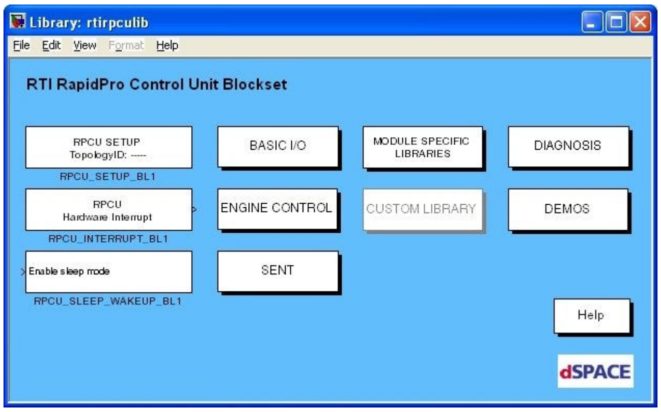
The RapidPro Control Unit comes with dSPACE implementation software (RTI and RTLib), which extends the I/O functionality of MicroAutoBox II or a DS1007 modular system with DS4121.

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/goto?support for software updates and patches. In addition, dSPACE recommends to check the status of your firmware periodically and to update the firmware to the latest version. To simplify the update procedure, dSPACE provides a specific RapidPro Firmware Update Tool, refer to How to Update RapidPro Firmware (RapidPro System Hardware Installation Guide 📖).

RTI

The real-time interface (RTI) makes it possible to implement the features of the Control Unit by adding graphical blocks from the RapidPro Control Unit Blockset (RTI RPCU Blockset) to your Simulink® model. The code which is created by MathWorks® Real-Time Workshop® is based on the RTLib functions.



RTLib

The Real-Time Library (RTLib) provides C functions for each implementation feature of the Control Unit, like A/D conversion, frequency measurement or injection and ignition pulse generation. You can use it for handcoded applications or for S-functions which you want to integrate in your Simulink model.

Note

Because of the modular concept of the RapidPro hardware, you can use only implementation features that are supported by the signal conditioning modules and/or power stage modules installed on the units of your RapidPro system.
For example, if you want to program A/D conversion, the RapidPro system must contain modules providing analog input channels. If no suitable modules are available, you cannot use the A/D conversion feature of the RapidPro Control Unit.

Related topics

Basics

Specifics of the RapidPro System

Objective

The architecture of a RapidPro system differs from other dSPACE hardware. Its high modularity affects implementation work and hardware handling.

Where to go from here

Information in this section

Modularity of RapidPro Hardware

Objective

A RapidPro system is built up individually by choosing different RapidPro units and modules which are required for your prototyping purpose.

Modules for Control Unit

The Control Unit provides 6 slots for signal conditioning modules (SC modules), for example:

SC Module	Description
SC-AI 4/1	Analog input module with 4 channels
SC-AI 10/1	Analog input module with 10 channels (requires 2 slots)
SC-DI 8/1	Digital input module with 8 channels

SC Module	Description
SC-DO 8/1	Digital output module with 8 channels
...	...

The slots can be equipped with up to 6 SC modules of the same type, or any combination of types. If your RapidPro system only contains the RapidPro Control Unit with the above mentioned modules, you can implement slave applications which require up to 30 analog input channels, for example, for A/D conversion, or 48 digital input or output channels, for example, for PWM signal generation and measurement.

Modules for SC Unit

If your application requires further input or output channels, you have to add one or more SC Units to your RapidPro system. An SC Unit provides 8 slots for signal conditioning modules. It can be equipped individually with the same modules as are available for the Control Unit.

Modules for Power Unit

If your application requires signals to drive actuators and loads, you have to add one or more Power Units to your RapidPro system. A Power Unit provides 6 slots for power stage modules (PS modules), for example:

PS Module	Description
PS-FBD 2/1	Full-bridge driver module with 2 channels
PS-LSD 6/1	Low-side driver module with 6 channels
PS-HSD 6/1	High-side driver module with 6 channels
...	...

For further hardware information on the RapidPro modules and units, refer to Hardware Overview (RapidPro System Hardware Reference 📖).

Signal Mapping to I/O Pins

Objective

Unlike other dSPACE hardware, RapidPro hardware does not have dedicated I/O mapping. The signal mapping to the I/O pins depends on various factors, for example, the modules which are installed in your RapidPro system. You can get the I/O mapping and pinout information of your hardware via ConfigurationDesk for RapidPro.

I/O mapping

Normally, your hardware provides static I/O mapping. That means that each pin of an I/O connector is reserved for one specific purpose. A microprocessor signal is directly connected to the corresponding I/O pin.

With RapidPro, all analog and digital signals to be processed by the microprocessor first need connecting to a corresponding signal conditioning or power stage module. Module types and the slots on which modules are installed can differ between the RapidPro systems because of their hardware modularity. The route of a signal from the I/O connector via the SC or PS module to the microprocessor is controlled by routing code.

Signal routing

Theoretically, the digital and analog routers make it possible to mount any type of SC module on any SC slot on the carrier board. In practice, the routers are controlled by a standard routing code, which allows a restricted selection of slot positions. For further information, refer to Restrictions on Changing SC/PS/COM Modules (RapidPro System Hardware Installation Guide 🕮).

Note

If you rearrange the modules on a carrier board, and the routing code of your system does not match the modified topology, the routing code has to be modified by dSPACE. For further information, refer to Effects of Changing Components of a RapidPro System on page 23.

RapidPro System Information Required for Implementation with RTI and RTLib

Objective

Because of the dependencies of usable software features and installed RapidPro hardware, you need specific information about the hardware when you implement a model. The required information about your individual RapidPro system is stored in the hardware topology file for RTI (*.hwt).

Contents of the HWT file

The hardware topology file for RTI (HWT file) contains:

- Topology ID

  This number indicates the hardware topology of your RapidPro system. Systems with identical hardware and routing code can be identified by the same topology ID. RTI and RTLib use this topology ID to verify the compatibility of the RapidPro system with your application.
- Hardware topology description of your RapidPro system

  The topology of your RapidPro system depends on the installed types of units and modules, and the slots the modules are installed on.
- I/O mapping information

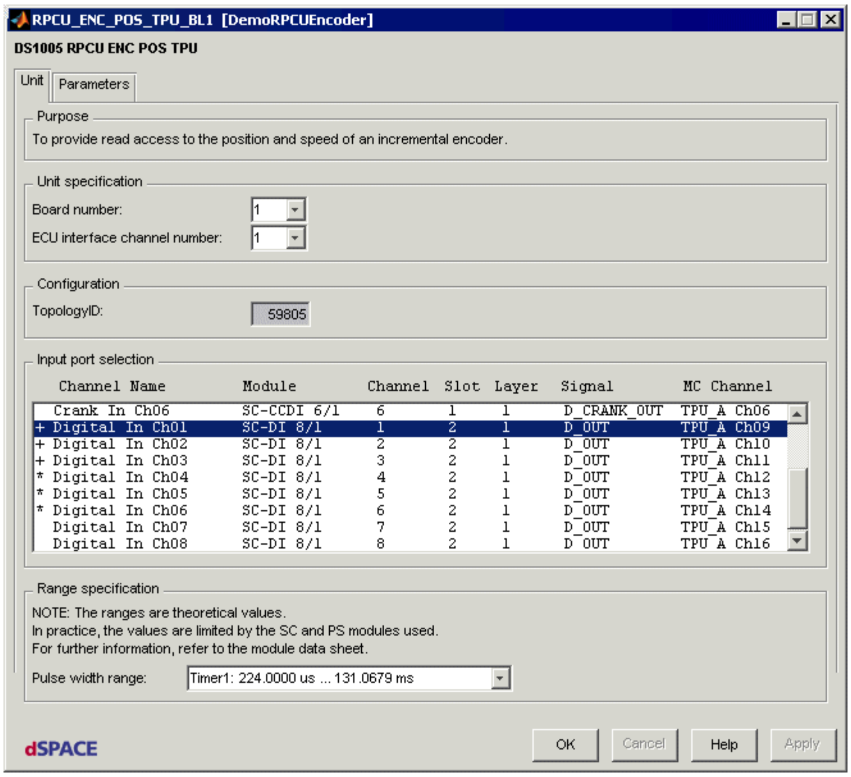
  The I/O mapping also depends on the hardware topology. It contains the I/O pin names, the signal conditioning modules used, and the channel names. You can edit the channel name via ConfigurationDesk for RapidPro to apply user-specific channel names. For example, the default label *analog input channel 30* can be renamed *BrakeSensor* to identify it as the brake sensor signal in your model. For detailed instructions, refer to the *ConfigurationDesk for RapidPro - Guide*.

Note

The hardware topology file for RTI comes with your RapidPro system. When you modify your system, you must generate a new hardware topology file using ConfigurationDesk for RapidPro. You should not make modifications which are not supported by the standard routing code. For further information on system modifications, refer to Effects of Changing Components of a RapidPro System on page 23, and for detailed instructions on generating the hardware topology file for RTI, refer to Exporting RapidPro Hardware Data (ConfigurationDesk for RapidPro - Guide 📖).

Implementing with RTI

After you have specified the hardware topology file in the setup block of the RTI RPCU Blockset, this information is available to all the RTI blocks in your model. See also How to Import the Hardware Topology to a Model on page 25. If you must select channels for an RTI block, the corresponding RTI block dialog displays a filtered view of the available channels. You can select only channels which support the specified function. For example, the Unit page of the Encoder block lists only the channels of installed modules which provide digital TPU inputs.



RPCU_ENC_POS_TPU_BL1 [DemoRPCUEncoder]

DS1005 RPCU ENC POS TPU

Unit | Parameters

Purpose

To provide read access to the position and speed of an incremental encoder.

Unit specification

Board number: 1

ECU interface channel number: 1

Configuration

TopologyID: 59805

Input port selection

```
  Channel Name        Module        Channel  Slot  Layer  Signal        MC Channel
  Crank In Ch06       SC-CCDI 6/1   6        1     1      D_CRANK_OUT   TPU_A Ch06
+ Digital In Ch01     SC-DI 8/1     1        2     1      D_OUT         TPU_A Ch09
+ Digital In Ch02     SC-DI 8/1     2        2     1      D_OUT         TPU_A Ch10
+ Digital In Ch03     SC-DI 8/1     3        2     1      D_OUT         TPU_A Ch11
* Digital In Ch04     SC-DI 8/1     4        2     1      D_OUT         TPU_A Ch12
* Digital In Ch05     SC-DI 8/1     5        2     1      D_OUT         TPU_A Ch13
* Digital In Ch06     SC-DI 8/1     6        2     1      D_OUT         TPU_A Ch14
  Digital In Ch07     SC-DI 8/1     7        2     1      D_OUT         TPU_A Ch15
  Digital In Ch08     SC-DI 8/1     8        2     1      D_OUT         TPU_A Ch16
```

Range specification

NOTE: The ranges are theoretical values.
In practice, the values are limited by the SC and PS modules used.
For further information, refer to the module data sheet.

Pulse width range: Timer1: 224.0000 us ... 131.0679 ms

Each list entry contains the channel name, the signal name, the name of the corresponding microcontroller channel, and a status flag. The meaning of the status flag is:

Status Flag	Meaning
*	Channel is already specified
+	Channel is specified for the current RTI block

Implementing with RTLib

With RTLib, you have no interactive support for channel selection as you have when you use RTI. You can get the necessary information about the available channels in the pinout information file that you can generate using ConfigurationDesk for RapidPro. This is a list in CSV format and corresponds to the HWT file. When you modify your system, you must not only generate a new HWT file using ConfigurationDesk for RapidPro, but also export a new pinout information file.

You have to announce the topology ID of the HWT file to your RTLib application to verify the compatibility of the RapidPro hardware. See also How to Import the Hardware Topology to a Model on page 25.

Effects of Changing Components of a RapidPro System

Objective

When you modify your RapidPro system, there are different effects depending on the changes you made.

Topology ID indicates modification type

Most modifications require the generation of a new hardware topology file for RTI (HWT file), but only some modifications result in a new topology ID. If the topology ID is the same after HWT file generation, the file is still valid for your implemented application. If the topology ID has changed, the channel mapping previously specified no longer matches your RapidPro hardware.

Modifications that change the topology ID

When you have modified the RapidPro system in such a way that generating a new HWT file results in a new topology ID, you have to reload the new HWT file to your model (see How to Import the Hardware Topology to a Model on page 25). Supposing you specified the channel mapping for your application before you changed a component of your RapidPro system, then the inport and outport channel selections are reset. That means that you have to specify the channel mapping of each RTI block again.

Note

With RTLib, there is no automatic reset of the specified channels when you change the topology ID in the initialization function. You must compare the already specified channels with the new generated pinout information file before executing the application to avoid hardware damage.

The topology ID changes when you:

- Replace a module with one of another type
- Change the slot of a module
- Add or remove a module
- Add or remove a unit to/from the system

Note

- Do not edit the HWT file manually. A modified file cannot be used with RTI.
- If you work with more than one RapidPro system, ensure that you have connected the hardware which corresponds to your application.

Modifications that do not change the topology ID

Some modifications have no effect on the topology ID when you generate a new HWT file. These are:

- Configuring a module
- Changing the channel names in ConfigurationDesk for RapidPro
- Replacing modules with ones of the same type but different hardware configurations
- Replacing a defective module with a module of the same type

Note

To make new channel names effective in your RTI model, you have to reload the hardware topology file (see How to Import the Hardware Topology to a Model on page 25). The file does not have to be reloaded for any of the other modifications listed above.

Recommendations

If you work with different RapidPro systems and models at the same time, you should store information about the required hardware in each model. For example, you can use a Simulink Model Info block for this.

How to Import the Hardware Topology to a Model

Objective

Before you can parameterize the RTI blocks of your controller model, you have to import the HWT file to the model. If you have modified the hardware topology information, you have to reimport the HWT file.

⚠ WARNING

If an HWT file with a new TopologyID number is opened, the Inport/Output Driver settings of all the blocks in the model are treated according to the hardware modification, as shown in the table below.

Hardware Modification	Driver Settings
Number of layers in the RapidPro stack did not change and the module used is still on the same place	Update: All drivers remain selected
Number of layers in the RapidPro stack changed	Reset: All drivers are set to <NOT SELECTED>
Layer, slot, or channel changed or Module has been substituted or I/O routing has changed	Reset: The drivers of the blocks concerned are set to <NOT SELECTED>

Using RTLib

When you handcode your application, the `dsrpcu_init` function needs only the topology ID as an input parameter. The function compares the specified topology ID with the topology ID of the connected RapidPro system. The contents of the HWT file are not imported to the RTLib application.

Preconditions

All block dialogs that display the Topology ID must be closed (except the RPCU_SETUP_BLx block).

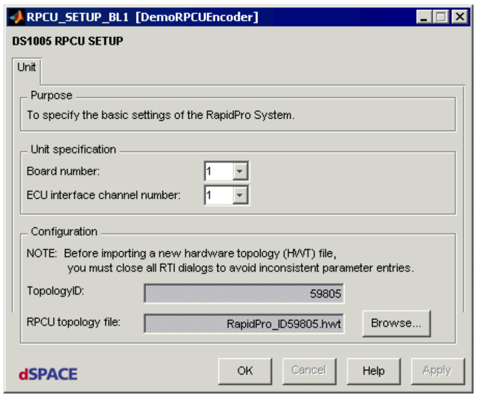
Method

To import the hardware topology to the model

1 Drag the main setup block (RPCU_SETUP_BLx block) into the model.

2 Double-click the block to open its dialog and select the required hardware topology file for RTI.

3 Click OK to close the dialog.


RPCU_SETUP_BL1 [DemoRPCUEncoder]

DS1005 RPCU SETUP

Unit

Purpose

To specify the basic settings of the RapidPro System.

Unit specification

Board number: 1

ECU interface channel number: 1

Configuration

NOTE: Before importing a new hardware topology (HWT) file, you must close all RTI dialogs to avoid inconsistent parameter entries.

TopologyID: 59805

RPCU topology file: RapidPro_ID59805.hwt

Result

The HWT file is parsed and its information is stored in the model. The RTI block dialogs use this information, for example, to create dynamically filtered channel selection lists. If you try to parameterize an RTI block without first specifying the HWT file you get an error message.

Note

When you download the compiled application to the hardware, the topology ID specified in the model must match the connected hardware, or you will get an error message.

Related topics

References

RPCU_SETUP_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Information for Implementing Prototyping Applications

Objective

The implementation features of the RapidPro Control Unit are I/O handling enhancements of the RCP system used. There are two use cases

- You want to create a new model from scratch based on the RapidPro system.
- You have an existing model running on the master of the RCP system without a slave connection.

Below there are some implementation notes for both use cases.

Where to go from here

Information in this section

Basic Information for Implementing with RapidPro Control Unit Blockset

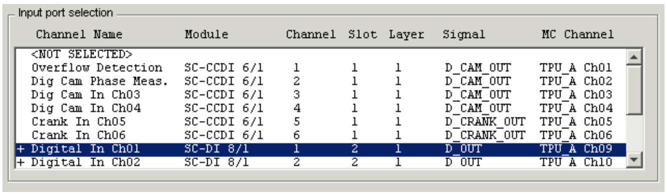
Objective

If you build a new model for a RapidPro system from scratch, the most convenient way is to use the RTI RapidPro Control Unit Blockset (RTI RPCU Blockset) to implement most of the features. This minimizes the processor load for the I/O handling on the master. In addition, the main feature of the RapidPro system is the integration of signal conditioning and power stage modules.

Available I/O features

The I/O features that you can use to build your model depend on the installed RapidPro hardware. You can add each block of the blockset to your model, but you can specify only I/O channels which are provided by the installed signal conditioning and power stage modules. The RTI block dialogs help you select the I/O channels by displaying a filtered view of all the available channels.

Selected channels are marked with an '*'. Channels that are selected for the current feature are marked with a '+’. If you want to deselect a specified channel, you must choose the 'Not selected' entry in the channel list.



Input port selection

Channel Name	Module	Channel	Slot	Layer	Signal	MC Channel
<NOT SELECTED>						
Overflow Detection	SC-CCDI 6/1	1	1	1	D_CAM_OUT	TPU_A Ch01
Dig Cam Phase Meas.	SC-CCDI 6/1	2	1	1	D_CAM_OUT	TPU_A Ch02
Dig Cam In Ch03	SC-CCDI 6/1	3	1	1	D_CAM_OUT	TPU_A Ch03
Dig Cam In Ch04	SC-CCDI 6/1	4	1	1	D_CAM_OUT	TPU_A Ch04
Crank In Ch05	SC-CCDI 6/1	5	1	1	D_CRANK_OUT	TPU_A Ch05
Crank In Ch06	SC-CCDI 6/1	6	1	1	D_CRANK_OUT	TPU_A Ch06
+ Digital In Ch01	SC-DI 8/1	1	2	1	D_OUT	TPU_A Ch09
+ Digital In Ch02	SC-DI 8/1	2	2	1	D_OUT	TPU_A Ch10

If the channel list contains no selectable channel, there is no hardware support for the feature. If your model contains a block without a specified channel, you will get a message when you execute the build process.

Note

The RapidPro hardware does not have to be connected to the RCP system for implementation, because the hardware system information is stored in the HWT file, which is imported to the model during model setup. If you download the application to the master and the connected RapidPro system has a different topology ID, you get an error message.

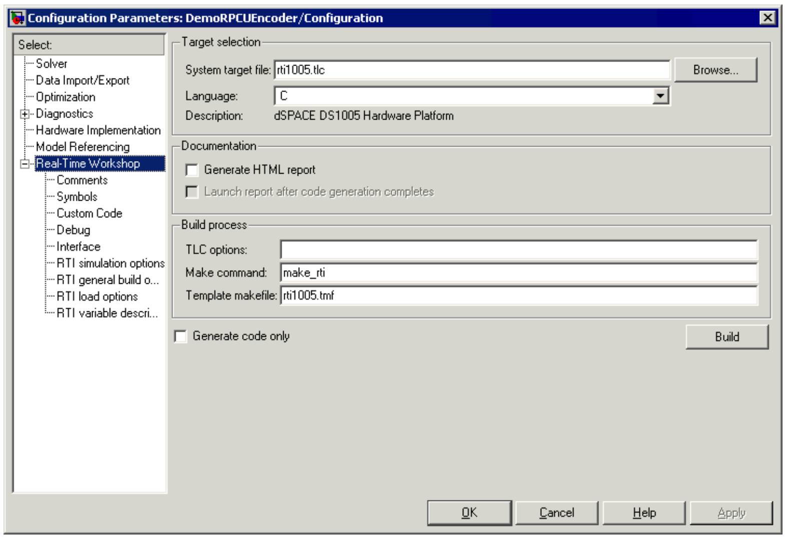
Integration of the RCP system

Integrating the RCP system, for example, MicroAutoBox II, into your model needs no additional effort, because you implement your model and build the application on the RCP system. The Control Unit acts only as a slave processor within the RCP environment.

You can build your model exclusively with RTI blocks of the RPCU Blockset. You do not need any RTI blocks from the RCP system's blockset in your model. If you do not include an RCP's RTI block, the model can be used for both RCP systems, MicroAutoBox II and DS1007 modular system. You only have to change the target-specific settings in Real-Time Workshop for building the application.

Note

The board-specific settings in the RTI blocks must match. The MicroAutoBox II module number must be the same as the DS1007 board number, and both must use the same ECU interface channel.



Target selection

System target file: rti1005.tlc

Language: C

Description: dSPACE DS1005 Hardware Platform

Documentation

☐ Generate HTML report

☐ Launch report after code generation completes

Build process

TLC options:

Make command: make_rti

Template makefile: rti1005.tmf

☐ Generate code only

Master-slave communication

Provided that the RapidPro system (slave) is ready for operation and connected to the RCP system (master), master-slave communication is initialized automatically when the application is started. Communication and data exchange are realized by two dual-port memories (DPMEM), one located at the master, the other at the slave. They are connected via LVDS module (see also Prototyping Systems with RapidPro Hardware on page 14). The LVDS link is connected to the ECU interface of the MicroAutoBox II or the ECU interface of a DS4121 board when using a PHS-bus-based system.

Connection to the host PC

When you want to download the compiled application to the master processor of your RCP system, the host PC and RCP system must be connected. You can use the same or another host PC to configure the RapidPro system. For further information, refer to Putting RapidPro Hardware into Operation (RapidPro System Hardware Installation Guide 📖).

Related topics

Basics

Overview of the RTI RapidPro Control Unit Blockset (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Information for Enhancing Existing Models

Objective

If a modification for an existing controller model requires additional I/O signals, which the RCP system is not able to provide, you can integrate the RapidPro Control Unit as a slave in your RCP environment. You can use the RapidPro system to increase the flexibility of the I/O features. All other I/O features are implemented with the RCP's RTI Blockset as before, for example, the RTI1401 Blockset.

Enhancements with new I/O features

If you add an I/O feature to a model that was not yet implemented, there are no conflicts with the existing RTI blocks, except for the required ECU interface.

Enhancements for implemented I/O features

When you replace an I/O feature which you implemented with the RCP's RTI Blockset with blocks from the RTI RPCU Blockset, you have to modify *all* the RTI blocks that correspond to the I/O feature. These are RTI blocks which depend on each other.

Related topics

Basics

Overview of the I/O Libraries (MicroAutoBox II RTI Reference 🕮)

Technical Details of the RapidPro Control Unit

Objective

Most characteristics of the I/O features that are provided by the Control Unit depend on the installed microprocessor. It is therefore useful to know some technical details of the MPC565 with its three time processor units (TPU), the two queued A/D converters (QADC) and the modular I/O system (MIOS). Further implementation-relevant hardware components are installed, for example, a programmable logic device (PLD). This is programmed by dSPACE and can be used for bit I/O handling.

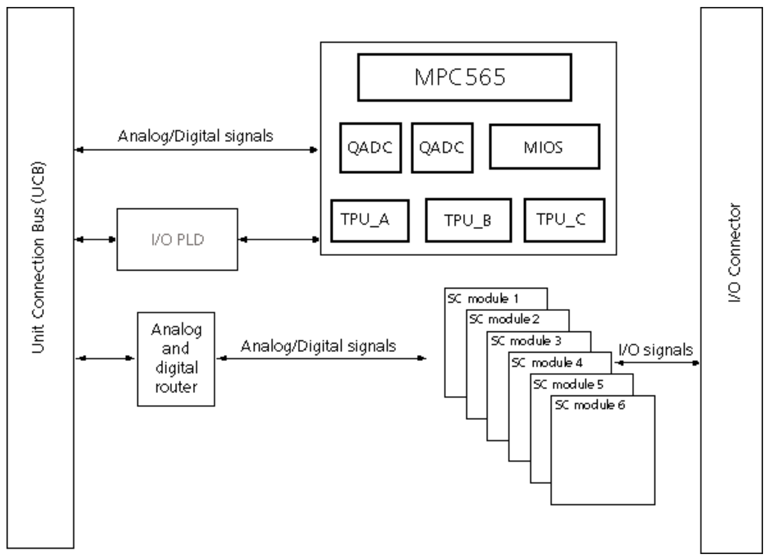
Where to go from here

Information in this section

Overview of the MPC565

Objective

The RapidPro Control Unit comes with a Motorola MPC565 microprocessor. It is installed on a microprocessor module with further components relevant for implementation.

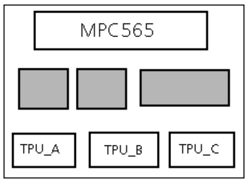


It has the following characteristics:

Feature	Characteristic
Processor clock rate	56 MHz
Main memory[1)]	▪ 4 MB external SRAM ▪ 16 MB external flash memory
Time processor units	3 TPUs with 16 channels each (For further details, see General Characteristics of a TPU on page 33)
Modular I/O system	MIOS with 22 channels (For further details, see General Characteristics of the MIOS on page 34)
A/D converters	2 queued A/D converters (QADC): ▪ Internal sample/hold ▪ 40 A/D converter channels ▪ 10-bit resolution ▪ 4 µs conversion time per channel
I/O PLD[2)]	40 bit I/O channels
Exchangeable transceiver	▪ 3 CAN interfaces ▪ 4 serial interfaces (i.e., RS232 or LIN)
Further external connections	Serial peripheral interface (SPI)

[1)] SRAM and flash memory does not belong to the MPC565. They are additional external hardware components.
[2)] The I/O PLD does not belong to the MPC565. It is an additional external hardware component.

General Characteristics of a TPU

Objective	A time processor unit (TPU) is a semi-autonomous microcontroller designed for timing control. It can operate simultaneously with the CPU with a minimum of intervention. Data can be stored locally in a dual-port memory. You can identify the functions which are executed on a TPU by the term *TPU* in RTI block names and RTLib function names. It can be used for most of the Control Unit features, like signal generation and signal measurement. 
Number of channels	Each of the three TPUs provides 16 independent timer channels. When you use a multi-channel feature, the required channels must be available on one TPU in sequential order.
Workload on a TPU	Each TPU has one execution unit for its 16 TPU channels. If all channels of a TPU execute a function, the TPU workload increases. This can affect items such as the frequency range for frequency measurement. The maximum frequency to be measured is smaller than the specified value.
Timer	The TPU timing I/O functions can be controlled via two different time bases, the first derived from the system clock, the second from the system clock or from the angular computation unit (ACU, see also Processing the Crankshaft Signal on page 147). The required timer resolution is realized by prescaling the time base value. For further information, refer to Timer Setup on page 53.
Request priority	Each task is determined internally by a TPU scheduler. The priority of a task can be specified for each channel as *low*, *middle,* or *high*. A priority-scheduling mechanism uses the priority and the channel number to decide which request will be serviced first. If you specified the same priority for all TPU channels, TPU channel 1 has the highest priority, and TPU channel 16 has the lowest priority. **Note** ▪ With RTI, the priority settings cannot be configured. They are set internally to a fixed value. ▪ With RTLib, you can specify the priority for each channel separately by setting the priority parameter to the required value.

Related topics

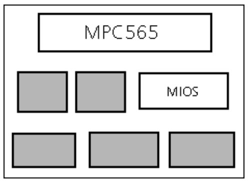
Basics

Comparing TPU and MIOS....................35

General Characteristics of the MIOS

Objective

The modular I/O subsystem (MIOS) of the microprocessor consists of a library of I/O and timer functions, i.e., a module for PWM signal generation. The MIOS provides channels which are especially suited to generating and measuring high-frequency signals. You can identify the functions which are executed on the MIOS by the term *MIOS* in RTI block names and RTLib function names.



Number of channels

The modular I/O system provides 12 channels for PWM signal generation.

Workload on the MIOS

Each MIOS channel works autonomously. The workload of the MIOS is therefore independent of the number of channels used. There is no negative effect on the frequency ranges for PWM signal generation.

Prescaling

Each PWM channel has its own prescaler to achieve the required resolution. For further information, refer to Timer Setup on page 53.

Related topics

Basics

Comparing TPU and MIOS....................35

Comparing TPU and MIOS

Objective

Some I/O features of the Control Unit can be implemented on a TPU and on the MIOS. Both components provide basically the same characteristics, but in some cases the differences may be relevant to your decision.

Comparison of the main characteristics

The main characteristics of TPU and MIOS are listed below as a decision aid:

Characteristics	TPU	MIOS
Number of channels	16 per TPU (a total of 48)	22 (12 for PWM signal generation)
Calculation unit	1 per TPU	1 per channel
Timer[1)	2 timers	Each PWM channel has its own prescaler.
Prescaler values[1)	Timer 1: 2 … 448 (13 steps) Timer 2: 8 … 120 (7 steps)	▪ Using RTI: Global: 3 Channel: 1 … 256 (9 steps, a power of 2) ▪ Using RTLib: Global: 2 … 16 (4 steps) Channel: 1 … 256 (256 steps)
Frequency ranges for 1-phase PWM signal generation[2)	Min.: 3.82 Hz … 4.46 kHz Max.: 854.5 Hz … 1 MHz	Min.: 0.21 Hz … 6.84 kHz Max.: 427.25 Hz … 14 MHz[3)
Interrupt support	Depends on the specific I/O feature	Depends on the specific I/O feature

1) For further information, refer to Timer Setup.
2) For further information, refer to Frequency Ranges for 1-Phase PWM Signal Generation.
3) This is a theoretical maximum value, because there is no RapidPro hardware module available at the moment that supports this frequency range.

Generally, you should prefer the MIOS functions because the frequency ranges are independent of the number of channels used. You also do not block TPU channels which are required for engine control features. However, a specific application may have requirements which are fulfilled by the TPU functions.

MIOS usage

The corresponding MIOS function should be used if:

- The function can handle the required frequency range.
- You need a constant frequency range independent from the number of channels used.
- You need multiple PWM channels to start synchronously.

TPU usage

The TPU functions should be used if:

- The function can handle the required frequency range.
- You need an interrupt after the generation of n PWM periods.
- You need synchronous update of duty cycles using multi-channel PWM signal generation.
- You want to calculate the average value from a frequency or pulse width measurement.
- You need more PWM channels than provided by the MIOS.
- You have to specify channel priorities (only RTLib).

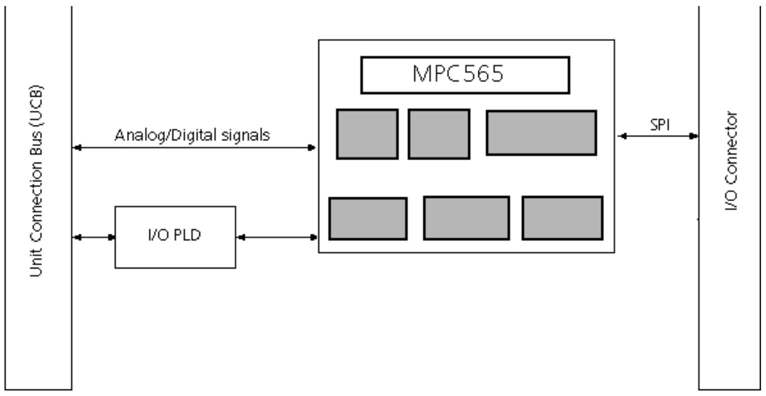
Related topics

Basics

General Characteristics of the I/O PLD

Objective

The programmable logic device (PLD) is used to enlarge the functionality of the MPC565.



Additional features

The I/O PLD is programmed by dSPACE. It provides functions for bit I/O, interrupt controlling, and the ACU that is used for engine control. While most of the features are used internally, you can access the PLD's bit I/O feature via RTI and RTLib. For further information, refer to Bit I/O on page 47.

Related topics

References

Bit I/O (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

A/D Conversion

Objective

The RapidPro Control Unit provides 2 parallel A/D converters that you can use for converting analog signals.

Where to go from here

Information in this section

Information in other sections

A/D Conversion (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Gives information about the RTI blocks concerned with A/D conversion.

A/D Conversion (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Gives information about the RTLib functions concerned with A/D conversion.

Basics on A/D Conversion

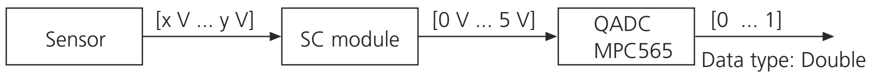
Objective

A/D conversion is an element of most applications in rapid control prototyping, because sensors, for example, for pressure or temperature, provide analog signals which have to be processed as digital signals.

Signal range

An incoming sensor signal is processed as follows:

A/D Conversion



1. Signal conditioning (SC module, for example, SC-AI 4/1):

   The incoming analog signal (bipolar or unipolar) is always transformed into the voltage range 0 V … 5 V, which is for compatibility with the voltage range of the analog inputs of the MPC565 microprocessor on the RapidPro Control Unit. You must use ConfigurationDesk for RapidPro to enter the **Input voltage range** of the incoming analog signal (x V … y V), refer to How to Change Parameters of User-Configurable Circuits (ConfigurationDesk for RapidPro - Guide 🕮).
2. A/D conversion (QADC MPC565):

   The A/D converter (QADC) of the MPC565 microprocessor on the RapidPro Control Unit converts the analog voltage range of 0 V … 5 V into a digital range of 0 … 1 (data type: Double). Refer to the I/O characteristics (MUX_ADC outport) of the RPCU_ADC_BLx.

Conversion configuration (standard and burst conversion)

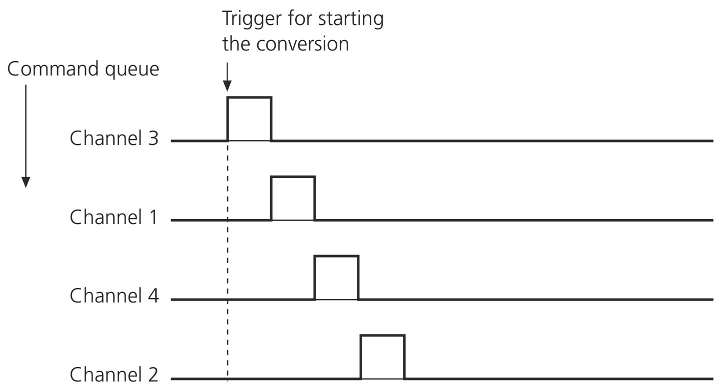
The two A/D converters execute a specified number of conversions starting with a trigger. You can configure the number and the sequence of the A/D channels to be used. You can specify the A/D channels in unsorted order, and the same channel can be set more than once. For example, you can specify the following channel list: 3, 9, 3, 36, 31.The A/D converters can be configured for *standard* A/D conversion and *burst* conversion:

- In standard conversion configuration, the trigger starts the A/D conversion according to the specified channel list. The A/D channels are switched from one channel to the next after completing a conversion. There is no additional trigger required to start the conversion on the current channel.
- In burst conversion configuration, the trigger starts the first specified A/D channel from the channel list for a specified number of conversions (you can specify 64 or 128 samples). The next trigger starts the next A/D channel on the channel list.

Both A/D converters can be configured independently. For further information on the conversion configurations, refer to Comparison of Standard and Burst A/D Conversion on page 45.

Conversion command queue

Both A/D converters have their own conversion command queue. The A/D conversion channels you have specified in the corresponding RTLib function or RTI block are written to the command queue, which assigns the execution order of the conversions.



Conversion modes

Regardless of which conversion configuration is specified (standard or burst conversion), you can choose between the two A/D conversion modes:

- Single mode

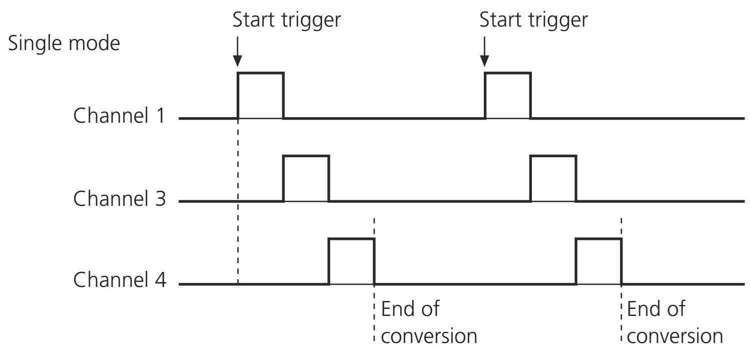
  The execution of the conversions specified in the command queue starts with a trigger event. The A/D converter stops at the end of conversion, which is reached when the end of the conversion command queue is processed. The result becomes available for the RCP system as follows:

  - The RapidPro system immediately sends the result to the RCP system, or
  - The RapidPro system buffers the result. After the number of results to be buffered is completed, the RapidPro system sends the buffered results to the RCP system all at once.

Note

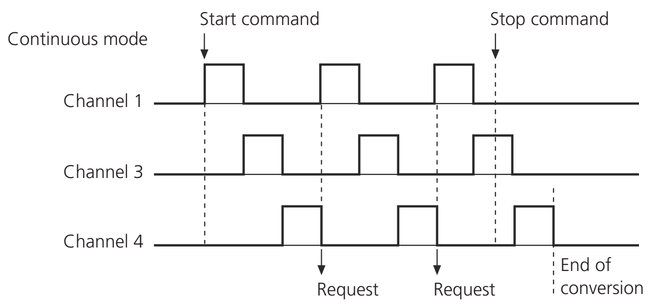
The RapidPro system can buffer up to 128 conversion results for each A/D conversion channel. The **Buffer conversion results** option reduces the CPU load of the RapidPro system and the master. However, it reduces the number of A/D conversion channels a command queue can hold, 12 instead of 20 for each A/D converter.

If you enabled the interrupt generation, the end-of-conversion interrupt is set after the data has been written to the DPMEM, and the master (MicroAutoBox II or DS1007) is triggered to read the conversion result from the DPMEM using the corresponding RTI block or RTLib function. The A/D converter waits for the next trigger event to start the next conversion. See also the picture below.



- Continuous mode

  The execution of the conversions in the conversion command queue starts with a start command in your application. A/D conversion continues until a stop command is called in your application. In continuous mode, the RTI ADC block provides an additional inport to start and stop conversion. You cannot start conversion using one of the triggers. The *end-of-conversion* interrupt is disabled. The converted values are available at the end of conversion.



Each time a QADC has serviced all A/D channels in the queue, it sends a request to the MPC565. The MPC565 starts a task, and the conversion results of the serviced A/D channels are transferred from the QADC via the MPC565 to the master at one go.

Note

The MPC565 is not stressed by the A/D conversions themselves (they are performed by the QADCs autonomously) but by the task of sending the A/D conversion results to the master. Specifying more A/D conversion channels increases the time between two consecutive data transfers to the master and reduces the stress on the MPC565 due to task handling. It is more efficient to send a big data packet after a long time period than to send small data packets very often because of the task overhead. Thus, if you use the continuous mode, you should specify at least 10 A/D channels in the queue to avoid a task overrun on the RapidPro Control Unit.

Trigger sources

The start of A/D conversion can be triggered by the following interrupts directly on the slave, without using a triggered subsystem on the master:

- PWM interrupt

  If your model contains an RPCU_PWM_TPU block, you can select the PWM mode in the ADC block dialog. In the PWM block dialog, you can specify the number of periods after which an interrupt is to be generated. This is a slave interrupt that can trigger the start of the A/D converters.
- Angle-based interrupt triggering the A/D converters

  If your model contains an RPCU_ANGLE_INT block for configuring angle-based interrupt generation, you can select the Engine mode in the ADC block dialog to trigger the A/D converter by an angle-based interrupt. Your model must also contain the appropriate blocks from the engine control blocks. For further information, refer to Generating Angle-Based Interrupts on page 263.

  Note

  If you use the engine mode on both converters, the triggers for converter A and converter B must have a minimum angle distance, which you can calculate by the following formula:

  `Angle_min = n · RPM / 10000 [°]`

  Where n = number of A/D channels specified in the queue. For A/D burst conversion: n = 32.
- External trigger

  A falling or rising edge of an external signal starts the conversion, for example, the angle-based trigger signal provided by a RPCU_CRANK_SETUP_BLx block (for details on the specification, refer to Advanced Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)).
- Sample base rate

  A request from the master (MicroAutoBox II or DS1007) starts the conversion. The A/D conversion is synchronized using the sample base rate of the model.

Interrupt on end of A/D conversion

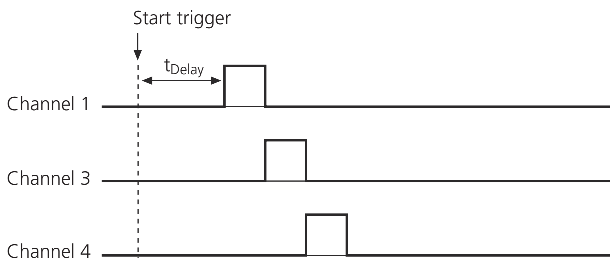
Both converters provide an interrupt at the end of an A/D conversion, if they are configured for single mode. In continuous mode, the generation of an end-of-conversion interrupt is not possible. For information on interrupt handling, see Interrupts Provided by the RapidPro Control Unit on page 349.

Time delay

In single conversion mode, you can specify the time interval between trigger event and conversion start. You can set the time delay using one of 11 predefined stages, which cover the range 45 µs … 47 ms.

Note

The time delay cannot be used if you have specified the engine mode, an external trigger or the continuous mode for A/D conversion.



Characteristics

The ADC unit of the Control Unit's microprocessor provides 2 queued A/D converters (QADC) with the following characteristics:

Characteristics	QADC
Number of analog input channels	40 Up to 20 on each converter (12, if the conversion results are buffered), as a combination of all 40 channels, for example, ADC channels 3, 9, 3, 36, 31 can be specified for A/D converter A
Resolution	10-bit
Conversion time	4 µs typical per channel
Buffer mechanism	Internal sample/hold

All other technical characteristics, for example, the input voltage range, depend on the signal conditioning module used.

Applicable modules

To use the A/D conversion feature, the RapidPro system must contain at least one of the following modules:

- SC-AI 4/1
- SC-AI 10/1
- Any other module providing analog signals

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 🕮).

RTI/RTLib support

You can access the A/D conversion features via the RTI RPCU Blockset and RTLib functions. For details, see:

- A/D Conversion (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- A/D Conversion (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

Basics

Comparison of Standard and Burst A/D Conversion.......45
End-of-Conversion Interrupt.......353

HowTos

How to Change Parameters of User-Configurable Circuits (ConfigurationDesk for RapidPro - Guide 🕮)

Comparison of Standard and Burst A/D Conversion

Objective

The two A/D converters of the RapidPro Control Unit can be configured for A/D standard conversion and A/D burst conversion.

A/D standard conversion

In the A/D standard conversion configuration, a trigger starts the execution of the command queue. Each specified A/D channel is converted once. The end of conversion is reached when all A/D channels in the queue have been converted.

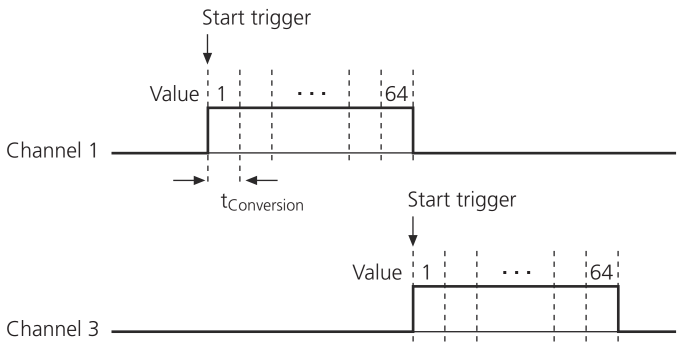
A/D burst conversion

In the A/D burst conversion configuration, a trigger starts the conversion on one channel. The burst size specifies the number of conversions to be performed for the triggered A/D channel. You can choose 64 or 128 as the burst size. The result is a sequence of conversion values. At the end of conversion, the next A/D channel from the channel list is activated and waits for the next trigger to start A/D conversion in burst mode.

Note

A/D burst conversions are performed with the fastest sample rate (conversion time), which is typically 4 µs (refer to MC-MPC565 1/1 Module (RapidPro System Hardware Reference 📖)). However, the signal frequency must not exceed the module-specific limit (cutoff or input frequency), refer to SC-AI 4/1 Module (RapidPro System Hardware Reference 📖) and SC-AI 10/1 Module (RapidPro System Hardware Reference 📖).

The following figure shows a burst conversion in single conversion mode on 2 channels with a burst size of 64.



Interrupt behavior

Interrupt generation is independent of the conversion configuration, but depends on the conversion mode. You can enable interrupt generation only if you specified the single conversion mode. In standard A/D conversion configuration, the end-of-conversion interrupt is generated if the conversion command queue has been executed. In burst A/D conversion configuration, the end-of-conversion interrupt is generated if a burst has been executed. For information, refer to Interrupts Provided by the RapidPro Control Unit on page 349.

Bit I/O

Objective

The RapidPro Control Unit provides functions to read and write digital signals. The bit I/O handling is supported by the three time processor units (TPUs) and a programmable logic device (I/O PLD).

Where to go from here

Information in this section

Information in other sections

Basics of Bit I/O

Characteristics

You can use functions on the TPU and on the PLD Bit I/O for the bit I/O access. Some characteristics of the two components differ:

Characteristics	**TPU A, B, C**	**PLD Bit I/O**
Number of I/O ports	48 (16 on each TPU)	40
Resolution	16 bit	8 bit (6 groups)
I/O access	Bit-wise	Group-wise (5 groups with 8 bits each) The bits of a group are set bit-wise.
Input/output	Input and output	Input and output
Power-up state	All digital I/O lines are disabled.	All digital I/O lines are configured as input.
Interrupt generation	Yes Interrupt generation on falling, rising or both edges of the input signal.	No

All other characteristics of bit I/O access, like the threshold voltage range, depend on the signal conditioning modules used.

Note

If you want to use the bit I/O features on a TPU, you must initialize the TPU timer beforehand. For information, refer to Timer Setup on page 53.

Applicable modules

To use the bit I/O feature, your RapidPro system must contain at least one of the following modules:

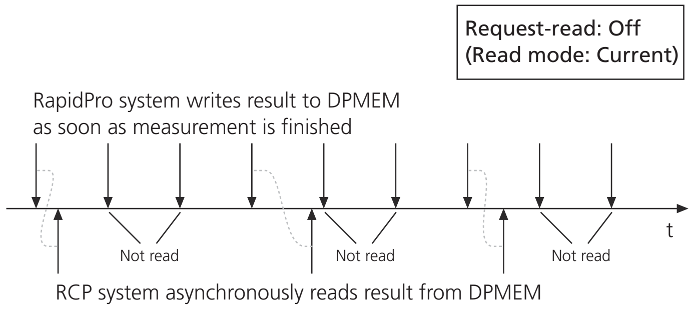
- SC-DI 8/1 for handling digital input signals
- SC-DO 8/1 for handling digital output signals
- Any other module providing digital signals

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 🕮).

Reading the measurement results

The RapidPro system offers several options for reading the measurement results:

- (Request-read=Off) The RapidPro system writes the results to the DPMEM as soon as a measurement is finished. The RCP system reads the DPMEM asynchronously and retrieves the currently available data.


Request-read: Off
(Read mode: Current)
RapidPro system writes result to DPMEM
as soon as measurement is finished
t
Not read
Not read
Not read
RCP system asynchronously reads result from DPMEM

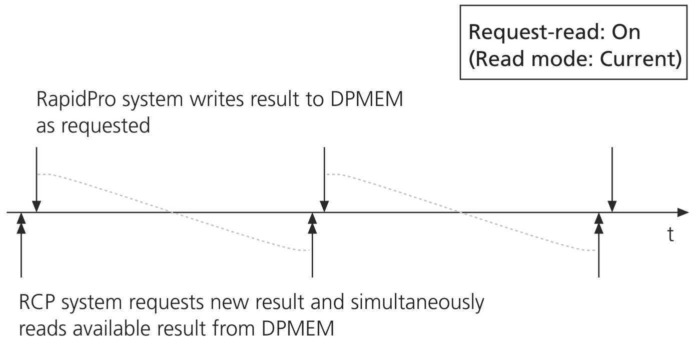
Note

If the frequency of the measured signal is high, the RapidPro system can be overloaded due to writing data to the DPMEM. Additionally, some results are written to the DPMEM but not read by the RCP system.

- (Request-read=On) The RapidPro system writes a new result to the DPMEM only when it has been requested by the RCP system.This is helpful if the frequency of the measured signal is high, as it avoids an unnecessary load on the CPU of the RapidPro system.

  The following two suboptions are possible:

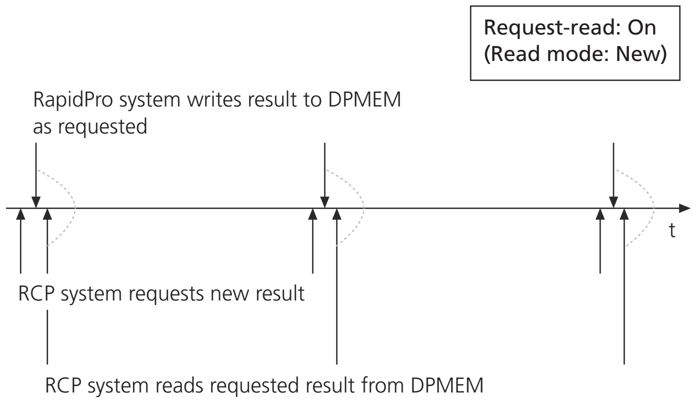
  - (Read mode="Current") The RCP system requests the RapidPro system to write a new result to the DPMEM but simultaneously reads the data that is currently available in the DPMEM.


Request-read: On
(Read mode: Current)
RapidPro system writes result to DPMEM
as requested
t
RCP system requests new result and simultaneously
reads available result from DPMEM

Note

As a rule, the data read by the RCP system represents the last but one sample step.

  - (Read mode="New") The RCP system requests the RapidPro system to write a new result to the DPMEM and waits for the requested data. The RapidPro system writes the requested data to the DPMEM. The RCP system reads the requested data from the DPMEM.


Request-read: On
(Read mode: New)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result
RCP system reads requested result from DPMEM

Note

Model execution on the RCP system slows down if it takes too much time for the RapidPro system to write the requested data to the DPMEM, for example, if the load on the CPU of the RapidPro system is high.

RTI/RTLib support

You can access the bit I/O features via the RTI RPCU Blockset and RTLib functions. For details, see:

- Bit I/O (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- Bit I/O (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

References

Advanced Page (RPCU_BIT_IN_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Parameters Page (RPCU_BIT_IN_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Timing I/O

Objective

The RapidPro Control Unit provides timing I/O features that you can use to generate and measure signals, for example, pulse width modulated (PWM) and square-wave signals.

Where to go from here

Information in this section

Information in other sections

Timing I/O (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Gives information about the RTI blocks concerned with timing I/O features.

Timing I/O (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Gives information about the RTLib functions concerned with timing I/O features.

Timer Setup

Objective

A timer is used to control the execution of time-dependent functions, for example, PWM signal generation implemented in a real-time application. Timers are clocked by a time base that can be modified by prescaling the basic clock rate. The prescaling values and the timer prescaling methods differ between TPU and MIOS.

Where to go from here

Information in this section

Information in other sections

RPCU_TIMER_SETUP_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
To configure the TPU timer settings.

Setup (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
To make the hardware topology known to the RTI model and configure the TPU timer settings.

Setup (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

Timer Basics

Objective

Before you can use the timing I/O features of the RapidPro Control Unit, you have to set up the timer. This is done by specifying a clock source and a prescaling factor.

Basics on prescaling

To achieve a high resolution over a wide range of frequencies, for example, you can modify the clock rate that is used for generating or measuring signals. Because of quantization effects, there can be a difference between the generated and the desired frequency, especially for higher frequencies. You should therefore select the range with the best possible resolution. You can achieve this by choosing the smallest prescaling factor that provides the period range that fits the generated or measured signal.

For example, if you specify a higher prescaling factor, the minimum and maximum values of the period range are lower.

Prescaling on a TPU

TPU timers

Each TPU can be controlled via two 16-bit time bases, represented by time counter registers TCR1 and TCR2:

- The time base of timer 1 (TCR1) is derived from the system clock.
- The time base of timer 2 (TCR2) is derived from the system clock or from the angular computation unit (ACU), when you select timer 2 for use with the engine control features. For further information on the ACU, refer to Processing the Crankshaft Signal on page 147.

Clock Source	Clock Rate
System clock on TCR1	56 MHz
System clock on TCR2	56 MHz
ACU clock on TCR2	0.1° pulses

Both time bases can be scaled within predefined values. A wide timer resolution range can be specified by the prescaler values. These settings can be made separately for each TPU.

Predefined prescaler values of timer 1

The predefined prescaler values of timer 1 (TCR1) and the resulting resolutions are:

Predefined Symbol	Value	Prescaled Clock	Resolution
TPU_TCR1_PSCK_2	2	28 MHz	35.71 ns
TPU_TCR1_PSCK_4	4	14 MHz	71.43 ns
TPU_TCR1_PSCK_8	8	7 MHz	142.9 ns
TPU_TCR1_PSCK_14	14	4 MHz	0.25 µs
TPU_TCR1_PSCK_28	28	2 MHz	0.5 µs
TPU_TCR1_PSCK_42	42	1.333 MHz	0.75 µs
TPU_TCR1_PSCK_56	56	1 MHz	1 µs
TPU_TCR1_PSCK_84	84	0.667 MHz	1.5 µs

Predefined Symbol	Value	Prescaled Clock	Resolution
`TPU_TCR1_PSCK_112`	112	0.5 MHz	2 µs
`TPU_TCR1_PSCK_168`	168	0.333 MHz	3 µs
`TPU_TCR1_PSCK_224`	224	0.25 MHz	4 µs
`TPU_TCR1_PSCK_336`	336	0.167 MHz	6 µs
`TPU_TCR1_PSCK_448`	448	0.125 MHz	8 µs

Predefined prescaler values of timer 2

The predefined prescaler values of timer 2 (TCR2) and the resulting resolutions are:

Predefined Symbol	Value	Prescaled Clock	Resolution
`TPU_TCR2_PSCK_8`	8	7 MHz	142.9 ns
`TPU_TCR2_PSCK_16`	16	3.5 MHz	285.7 ns
`TPU_TCR2_PSCK_24`	24	2.333 MHz	428.6 ns
`TPU_TCR2_PSCK_32`	32	1.75 MHz	571.4 ns
`TPU_TCR2_PSCK_56`	56	1 MHz	1 µs
`TPU_TCR2_PSCK_64`	64	0.875 MHz	1.143 µs
`TPU_TCR2_PSCK_120`	120	0.467 MHz	2.143 µs

Note

If you want to use engine control features that require angle detection, you have to select timer 2. In this case, the specified prescaler value for timer 2 has no effect because the clock source changes from the system clock to the external input coming from the ACU. For further details, refer to Processing the Crankshaft Signal on page 147.

Range settings

The two time bases and the prescaler values are specified via the RPCU_TIMER_SETUP_TPU_BLx block. The timer setup affects the range settings, for example, for signal generation and measurement. In the TPU timing I/O RTI blocks, you can select only the timer to be used. The range settings themselves cannot be modified in the block dialogs.

When you use RTLib, prescaling is done in a similar way via the timer setup functions.

Related topics

Basics

HowTos

How to Specify TPU Timers

Objective

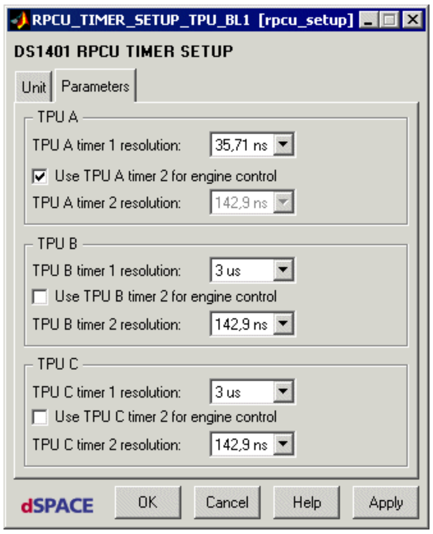
The features that are provided by the TPUs are timer-driven. To influence the timing behavior, for example, the frequency range of a signal measurement, you can specify different timer resolutions for the two available timers for each TPU. These settings are made globally using an RTI timer setup block in your model.

Method

To specify TPU timers

1. Drag the RPCU_TIMER_SETUP_TPU_BLx block into the model.
2. Double-click the block to open its dialog and switch to the Parameters page.
3. Specify the resolutions of timer 1 and timer 2 for each TPU you want to use in your model.

   If you implement engine control features, you have to activate use of timer 2 for engine control by selecting the corresponding checkbox for each required TPU. If you activate timer 2 for engine control, the time base is switched to the ACU clock. For further information, refer to Setting Up Crankshaft and Camshaft Signal Measurement on page 200.


RPCU_TIMER_SETUP_TPU_BL1 [rpcu_setup]

DS1401 RPCU TIMER SETUP

Unit | Parameters

TPU A
TPU A timer 1 resolution: 35,71 ns
☑ Use TPU A timer 2 for engine control
TPU A timer 2 resolution: 142,9 ns

TPU B
TPU B timer 1 resolution: 3 us
☐ Use TPU B timer 2 for engine control
TPU B timer 2 resolution: 142,9 ns

TPU C
TPU C timer 1 resolution: 3 us
☐ Use TPU C timer 2 for engine control
TPU C timer 2 resolution: 142,9 ns

Result

In the TPU timing I/O block dialogs, the selectable range settings of timer 1 and timer 2 correspond to the timer setup settings.

Using RTLib, you also have to initialize each TPU by configuring the prescaler settings of timer 1 and timer 2 before you can use TPU functions. For further information, refer to TPU Initialization (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮).

Prescaling on the MIOS

Objective

The modular I/O subsystem (MIOS) has its own timer. The resolution of the MIOS timing I/O features depends on the specified prescaling factor, which consists of a global prescaler value and a channel-specific prescaler value.

Calculating the prescaling factor

The total prescaling factor depends on the global prescaler value and the channel-specific prescaler value:

```
P_total = P_global · P_channel
```

```
P_global = 2 … 16 and P_channel = 1 … 256
```

Prescaling with RTI

With RTI, the timer resolution is automatically set by selecting the required period or frequency range in the corresponding block dialog.

The available prescaling factors are reduced to a range that covers the period or frequency ranges usually required:

- The global prescaler value is always set to P_global = 3.
- The channel-specific prescaler value is a power of 2 within the specified prescaler range (1, 2, 4, 8, … , 256).

The predefined prescaler values and the resulting resolutions are:

Prescaler Value	Prescaled Clock	Resolution
P_global = 3 (`DSRPCU_MIOS_GLOBAL_PSCK_3`)		
P_channel =		
1	18.67 MHz	53.6 ns
2	9.33 MHz	0.11 µs
4	4.67 MHz	0.21 µs
8	2.33 MHz	0.43 µs
16	1.17 MHz	0.86 µs
32	583.33 kHz	1.71 µs
64	291.67 kHz	3.43 µs
128	145.83 kHz	6.86 µs
256	72.92 kHz	13.71 µs

Prescaling with RTLib

With RTLib, the total prescaling factor is a combination of the global prescaler and the prescaler value that you can specify for each channel.

- Global prescaler value

  The global prescaler value is specified via the MIOS setup function. It divides the basic clock rate of the MIOS of 56 MHz. The predefined global prescaler values of the MIOS timer are:

Predefined Symbol	Value	Prescaled Clock
`DSRPCU_MIOS_GLOBAL_PSCK_2`	2	28 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_3`	3	18.67 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_4`	4	14 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_5`	5	11.2 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_6`	6	9.33 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_7`	7	8 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_8`	8	7 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_9`	9	6.22 MHz
`DSRPCU_MIOS_GLOBAL_PSCK_10`	10	5.6 MHz

Predefined Symbol	Value	Prescaled Clock
DSRPCU_MIOS_GLOBAL_PSCK_11	11	5.09 MHz
DSRPCU_MIOS_GLOBAL_PSCK_12	12	4.67 MHz
DSRPCU_MIOS_GLOBAL_PSCK_13	13	4.31 MHz
DSRPCU_MIOS_GLOBAL_PSCK_14	14	4 MHz
DSRPCU_MIOS_GLOBAL_PSCK_15	15	3.73 MHz
DSRPCU_MIOS_GLOBAL_PSCK_16	16	3.5 MHz

- Channel prescaler value

  You can specify the channel's prescaler value in the initialization functions for a MIOS timing I/O feature. The prescaler values of the PWM channels are in the range 1 … 256.

Period range calculation for PWM signal generation

The period range can be calculated using the following formula:

```
T_max = (65535 · P_global · (256 - P_channel)) / Clock_MPC565
```

```
T_min = (2 · P_global · (256 - P_channel)) / Clock_MPC565
```

```
Clock_MPC565 = 56 MHz
P_global = 2 … 16
P_channel = 0 … 255
```

Related topics

Basics

PWM Signal Generation

Objective

The RapidPro Control Unit provides timing I/O features that you can use to generate various PWM signals.

Where to go from here

Information in this section

Information in other sections

PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Information about the RTI blocks concerned with PWM signal generation.

PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Information about the RTLib functions concerned with PWM signal generation.

1-Phase PWM Signal Generation

Objective

The RapidPro Control Unit provides outputs for 1-phase PWM signal generation.

Where to go from here

Information in this section

Information in other sections

Basics of 1-Phase PWM Signal Generation

PWM signals

PWM signal generation is fundamental to many motor and motion control applications. A PWM signal is characterized by its period, duty cycle, and polarity.

PWM period

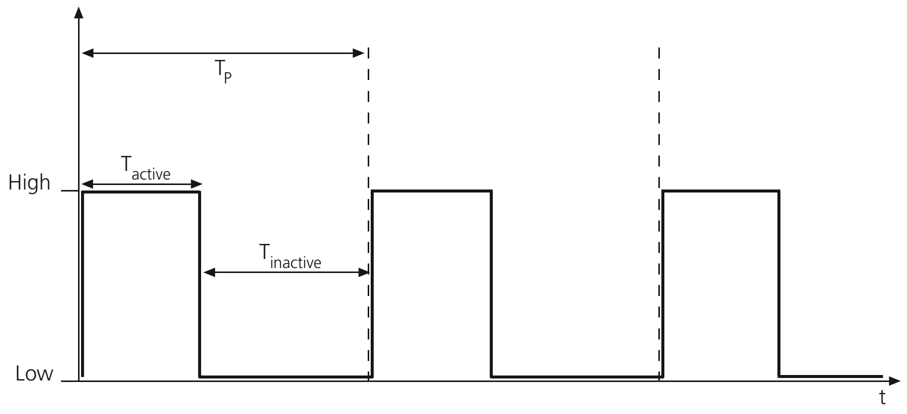
For PWM signals, you can specify the PWM period T_P (= T_{active}+ $T_{inactive}$) within a range that corresponds to the specified timer and prescaler value. For further information on prescaling, refer to Timer Setup on page 53. Each of the PWM output channels can have different values for the PWM period.

Note

The ranges are theoretical values. In practice, the values depend on the SC and PS modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference 🕮.

Duty cycle

You can specify the duty cycle, which is defined as d = T_{active}/ T_P. The available duty cycle range is 0 … 1 (0 … 100 %).



Note

If you generate a PWM signal at the upper limit of the specified period range, it is not guaranteed that the duty cycle can be changed in small steps, for example, steps of 1%. To ensure precise duty cycle updating, you can select another period range.

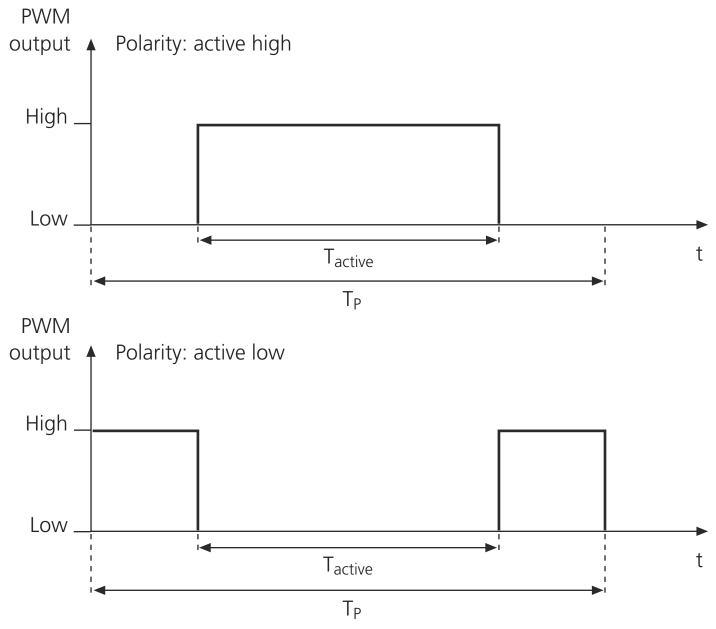
Polarity

With RTI, the polarity of the PWM outputs is specified by the configuration of the SC or PS modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference .

Note

With RTLib, the polarity is specified by a function parameter and the configuration of the SC and PS modules used. It can happen, that the module configuration using ConfigurationDesk for RapidPro inverts the specified signal polarity again.

The following illustration shows how the duty cycle (T_{active}/T_P ratio) and polarity are defined.



PWM signal generation using the Control Unit

1-phase PWM signal generation using the RapidPro Control Unit has the following characteristics:

- Run-time adjustable period
- Run-time adjustable duty cycle

1-phase PWM signal generation can be implemented on the MIOS and on the TPUs.

Overview of TPU/MIOS characteristics

The differences between signal generation on the TPU and on the MIOS are:

Characteristics	TPU A, B, C	MIOS
Channel number	1 … 16 on each TPU	1 … 12
Update mode	Update of duty cycle and period1)	▪ Update of duty cycle and period1) ▪ Update of duty cycle only
Synchronous start	No	Yes
Interrupt generation	Yes Interrupt rate within the range 1 … 256 periods	No
Prescaler range	The current prescaling factor depends on the prescaler setup of the specified time counter register.	The current prescaling factor depends on the channel-specific prescaler value.

Characteristics	TPU A, B, C	MIOS
Period range	Depends on the prescaler and the SC and PS modules used. Refer to Frequency Ranges for 1-Phase PWM Signal Generation on page 64.	

[1] Updating the duty cycle and period simultaneously using the MIOS PWM requires more MPC565 power than using the TPU PWM, because the MIOS triggers an interrupt on the MPC565 after each period update.

For further information on the TPU characteristics, refer to General Characteristics of a TPU on page 33.

For further information on the MIOS characteristics, refer to General Characteristics of the MIOS on page 34.

Applicable modules

To use the PWM signal generation feature, your RapidPro system must contain at least one of the following modules:

- SC-DO 8/1
- A PS module, if your RapidPro system contains a Power Unit
- Any other module providing digital output signals

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 🕮).

RTI/RTLib support

You can access the 1-phase PWM signal generation feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Frequency Ranges for 1-Phase PWM Signal Generation

Ranges on the TPU

The period range for signal generation on a TPU channel depends on the prescaler setting. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54.

Prescaler Value	f_{min}	f_{max}
TCR1		
TPU_TCR1_PSCK_2	854.5 Hz	1 MHz
TPU_TCR1_PSCK_4	427.3 Hz	500 kHz
TPU_TCR1_PSCK_8	213.7 Hz	250 kHz
TPU_TCR1_PSCK_14	122.1 Hz	142 kHz
TPU_TCR1_PSCK_28	61.1 Hz	71.4 kHz
TPU_TCR1_PSCK_42	40.7 Hz	47.6 kHz
TPU_TCR1_PSCK_56	30.6 Hz	35.7 kHz
TPU_TCR1_PSCK_84	20.4 Hz	23.8 kHz
TPU_TCR1_PSCK_112	15.3 Hz	17.8 kHz
TPU_TCR1_PSCK_168	10.2 Hz	11.9 kHz
TPU_TCR1_PSCK_224	7.63 Hz	8.92 kHz
TPU_TCR1_PSCK_336	5.09 Hz	5.95 kHz
TPU_TCR1_PSCK_448	3.82 Hz	4.46 kHz
TCR2		
TPU_TCR2_PSCK_8	213.7 Hz	250 kHz
TPU_TCR2_PSCK_16	106.9 Hz	125 kHz
TPU_TCR2_PSCK_24	71.3 Hz	83.3 kHz
TPU_TCR2_PSCK_32	53.5 Hz	62.5 kHz
TPU_TCR2_PSCK_56	30.6 Hz	35.7 kHz
TPU_TCR2_PSCK_64	26.8 Hz	31.2 kHz
TPU_TCR2_PSCK_120	14.3 Hz	16.6 kHz

Note

The described ranges are affected by the TPU workload. If the TPU executes 1-phase PWM signal generation on more than one channel, or executes additional functions, the maximum value might not be reached.

Ranges on the MIOS

The period range for PWM signal generation on the MIOS depends on the specified global prescaler value. The period and the corresponding resolution can be calculated using the following equations:

```
F_min = Clock_MPC565 / (65535 · P_global · (256 - P_channel))
```

```
F_max = Clock_MPC565 / (2 · P_global · (256 - P_channel))
```

```
T_max = (65535 · P_global · (256 - P_channel)) / Clock_MPC565
```

```
T_min = (2 · P_global · (256 - P_channel)) / Clock_MPC565
```

```
Clock_MPC565 = 56 MHz
P_global = 2 … 16
P_channel = 0 … 255
```

Note

Using RTI, the selectable period ranges are calculated with a global prescaler value of 3 and a channel-specific prescaler value, which is a power of 2. For further information, refer to Prescaling on the MIOS on page 57.

The following table shows some examples of the minimum and maximum values of the period ranges depending on the prescaler values:

Prescaler Value		f_{min}	f_{max}	T_{max}	T_{min}
Global Prescaler	**Channel Prescaler**				
2	0	1.67 Hz	54.69 kHz	0.599 s	18.28 µs
	255	427.25 Hz	14 MHz	2.34 ms	71.4 ns
3[1)]	0	1.11 Hz	36.458 kHz	898.7 ms	27.43 µs
	255	284.84 Hz	9.33 MHz	3.510 ms	107.2 ns
4	0	0.83 Hz	27.34 kHz	1.198 s	36.57 µs
	255	213.62 Hz	7 MHz	4.68 ms	142.85 ns
8	0	0.42 Hz	13.67 kHz	2.397 s	73.14 µs
	255	106.81 Hz	3.5 MHz	9.36 ms	285.71 ns
16	0	0.21 Hz	6.84 kHz	4.793 s	0.146 ms
	255	53.41 Hz	1.75 MHz	18.7 ms	571.4 ns

1) RTI uses only this global prescaler value.

Note

The ranges are theoretical values. In practice, the values depend on the SC and PS modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference .

Related topics

Basics

Edge-Aligned Multi-Channel PWM Signal Generation (MCPWM_EA)

Objective

The RapidPro Control Unit provides edge-aligned multi-channel PWM signal generation (MCPWM_EA). With this feature, you can generate 3 PWM signals of the same frequency, which are synchronized to their rising edges. The duty cycles of the 3 PWM signals can be adjusted separately during run time.

Where to go from here

Information in this section

Information in other sections

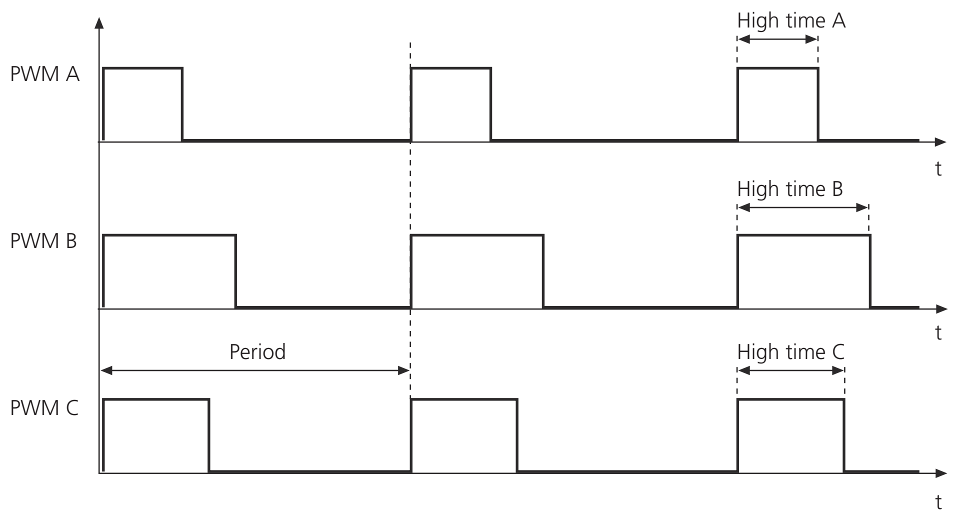
RPCU_MCPWM_EA_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
To generate edge-aligned PWM signals with variable duty cycles on a TPU channel.

PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Information about the RTLib functions concerned with PWM signal generation.

Basics of Edge-Aligned Multi-Channel PWM Signal Generation

Signal characteristics

The PWM signals are synchronized to the rising edges of the periods. The signals start at high level and switch to low level at different times according to their duty cycles.



Edge-aligned multi-channel PWM signal generation has the following characteristics:

- Run-time adjustable duty cycle (T_{high}/T_p ratio) on each channel
- Variable interrupt position

Interrupt position

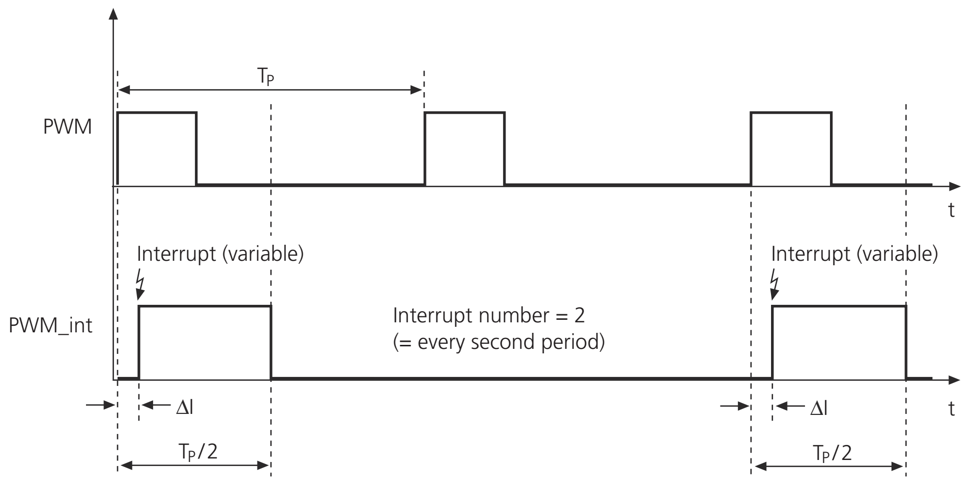
The interrupt position can be specified as fixed or variable. If you select interrupt generation with a fixed position, the interrupt is triggered at the rising edge of the PWM signals. If you select interrupt generation with variable positioning, the interrupt is triggered with a time delay ΔI after the rising edge, refer to the parameter **Interrupt delay** (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮). The specified time delay is valid for the entire model execution time. You can specify the number of PWM periods after which an interrupt is periodically generated. The range is 1 ... 256.

Note

For the use of variable interrupts, an additional PWM signal (see PWM_int in the illustration below) is generated which is specified as follows:

- Rising edge at the same position as the variable interrupt
  By specifying the number of PWM periods after which a variable interrupt is periodically generated, you also specify the number of PWM periods after which the additional PWM signal PWM_int is periodically generated.
- Falling edge at the middle of the period of the edge-aligned multi-channel PWM signal

Due to the master-slave communication it might happen that an interrupt reaches the master processor a few microseconds later than specified. The actual delay depends on your application.



Tip

If the interrupt channel (TPU channel 5) is routed as a digital output channel, you can use PWM_int as an external trigger source for A/D conversion.

Overview of TPU characteristics

The MCPWM_EA feature is supported only by the TPU. Its characteristics are:

Characteristics	TPU A, B, C
Channel number	Channel assignment must start with the 1st channel of a TPU. ▪ 4 successive channels are normally used ▪ 5 successive channels are used if you specify variable interrupt positioning
Update mode	Update of duty cycle
Interrupt generation	Yes
Interrupt mode	Interrupt can be specified as: ▪ Disabled ▪ Fixed position (on the rising edge) ▪ Variable position specified by a time delay
Prescaler range	The current prescaling factor depends on the prescaler setup of the specified time counter register.
Period range	Depends on the prescaler and the SC and PS modules used, refer to Frequency Ranges for Edge-Aligned Multi-Channel PWM Signal Generation on page 71.

Channel usage

You must specify the 1st channel of a TPU to configure the MCPWM_EA feature in your model. If channel 1 from TPUs A, B and C are already used by other blocks, it is not possible to implement this feature.

TPU channel 1 is used internally only. The 3 PWM output channels are allocated automatically. If you enable interrupt generation with variable interrupt positioning, a 5th channel is allocated for interrupt generation:

Channel Used	Meaning
TPU channel 1	Channel which you must specify in the RTI block or RTLib function. It is not used for output signal generation. It must always be the first channel of the TPU.
TPU channel 2	PWM output channel a (duty 1)
TPU channel 3	PWM output channel b (duty 2)
TPU channel 4	PWM output channel c (duty 3)
TPU channel 5 (opt.)	If variable interrupt positioning is enabled, this channel is used for the additionally generated PWM signal (PWM_int). Otherwise, this channel is not used.

Applicable modules

To use the edge-aligned multi-channel PWM signal generation feature, your RapidPro system must contain at least one of the following modules:

- SC-DO 8/1
- A PS module, if your RapidPro system contains a Power Unit
- Any other module providing digital output signals

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 📖).

RTI/RTLib support

You can access the edge-aligned multi-channel PWM signal generation feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

Basics

Frequency Ranges for Edge-Aligned Multi-Channel PWM Signal Generation

Ranges on the TPU

The period range for signal generation on a TPU channel depends on the prescaler setting. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54.

Prescaler Value	f_{min}	f_{max}	T_{max}	T_{min}
TCR1				
TPU_TCR1_PSCK_2	1709 Hz	200 kHz	0.585 ms	5.0 µs
TPU_TCR1_PSCK_4	854.5 Hz	100 kHz	1.17 ms	10 µs
TPU_TCR1_PSCK_8	427.3 Hz	50 kHz	2.34 ms	20 µs
TPU_TCR1_PSCK_14	244.2 Hz	28.5 kHz	4.09 ms	35.09 µs
TPU_TCR1_PSCK_28	122.1 Hz	14.2 kHz	8.19 ms	70.42 µs
TPU_TCR1_PSCK_42	81.4 Hz	9.52 kHz	12.28 ms	105.04 µs
TPU_TCR1_PSCK_56	61.1 Hz	7.14 kHz	16.37 ms	140.06 µs
TPU_TCR1_PSCK_84	40.7 Hz	4.76 kHz	24.57 ms	210.08 µs
TPU_TCR1_PSCK_112	30.6 Hz	3.57 kHz	32.68 ms	280.11 µs
TPU_TCR1_PSCK_168	20.4 Hz	2.38 kHz	49.02 ms	420.17 µs
TPU_TCR1_PSCK_224	15.3 Hz	1.78 kHz	65.36 ms	561.8 µs
TPU_TCR1_PSCK_336	10.2 Hz	1.19 kHz	98.04 ms	840.34 µs
TPU_TCR1_PSCK_448	7.63 Hz	892 Hz	131.06 ms	1.12 ms
TCR2				
TPU_TCR2_PSCK_8	427.3 Hz	50 kHz	2.34 ms	20.0 µs
TPU_TCR2_PSCK_16	213.6 Hz	25 kHz	4.68 ms	40.0 µs
TPU_TCR2_PSCK_24	142.4 Hz	16.6 kHz	7.02 ms	60.24 µs
TPU_TCR2_PSCK_32	106.8 Hz	12.5 kHz	9.36 ms	80.0 µs
TPU_TCR2_PSCK_56	61.0 Hz	7.14 kHz	16.38 ms	140.06 µs
TPU_TCR2_PSCK_64	53.4 Hz	6.25 kHz	18.72 ms	160.0 µs
TPU_TCR2_PSCK_120	28.5 Hz	3.33 kHz	35.11 ms	300.0 µs

Note

- The described ranges are affected by the TPU workload. If the TPU executes additional functions, the maximum frequency value might not be reached.
- The ranges are theoretical values. In practice, the values depend on the SC and PS modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference 🕮.

Related topics

Basics

Center-Aligned Multi-Channel PWM Signal Generation (MCPWM_CA)

Objective

The RapidPro Control Unit provides center-aligned multi-channel PWM signal generation (MCPWM_CA). This makes it possible to implement 3 PWM signals, which are synchronized to the middle of their high times. You can configure the MCPWM_CA feature to provide also the inverted PWM signals.

Where to go from here

Information in this section

Information in other sections

RPCU_MCPWM_CA_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
To generate center-aligned PWM signals with variable duty cycles on a TPU channel.

PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Information about the RTLib functions concerned with PWM signal generation.

Basics of Center-Aligned Multi-Channel PWM Signal Generation

Signal characteristics

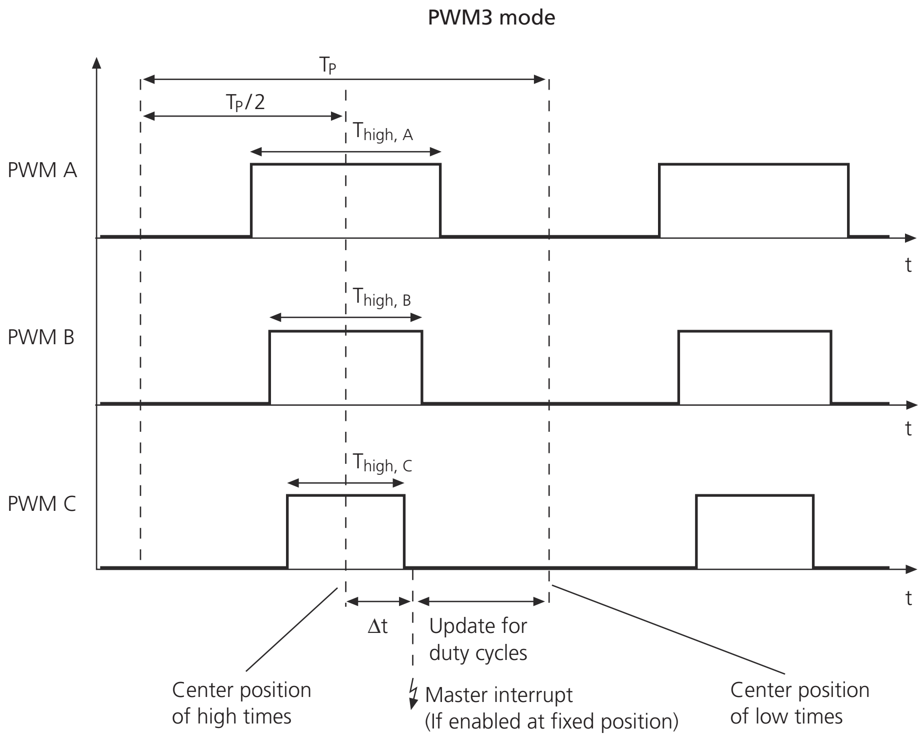
The PWM signals are synchronized to the middle of the periods. A period starts in the middle of a low level signal. The center-aligned mode enables you to use dead times (also called deadband).

Center-aligned multi-channel PWM signal generation has the following characteristics:

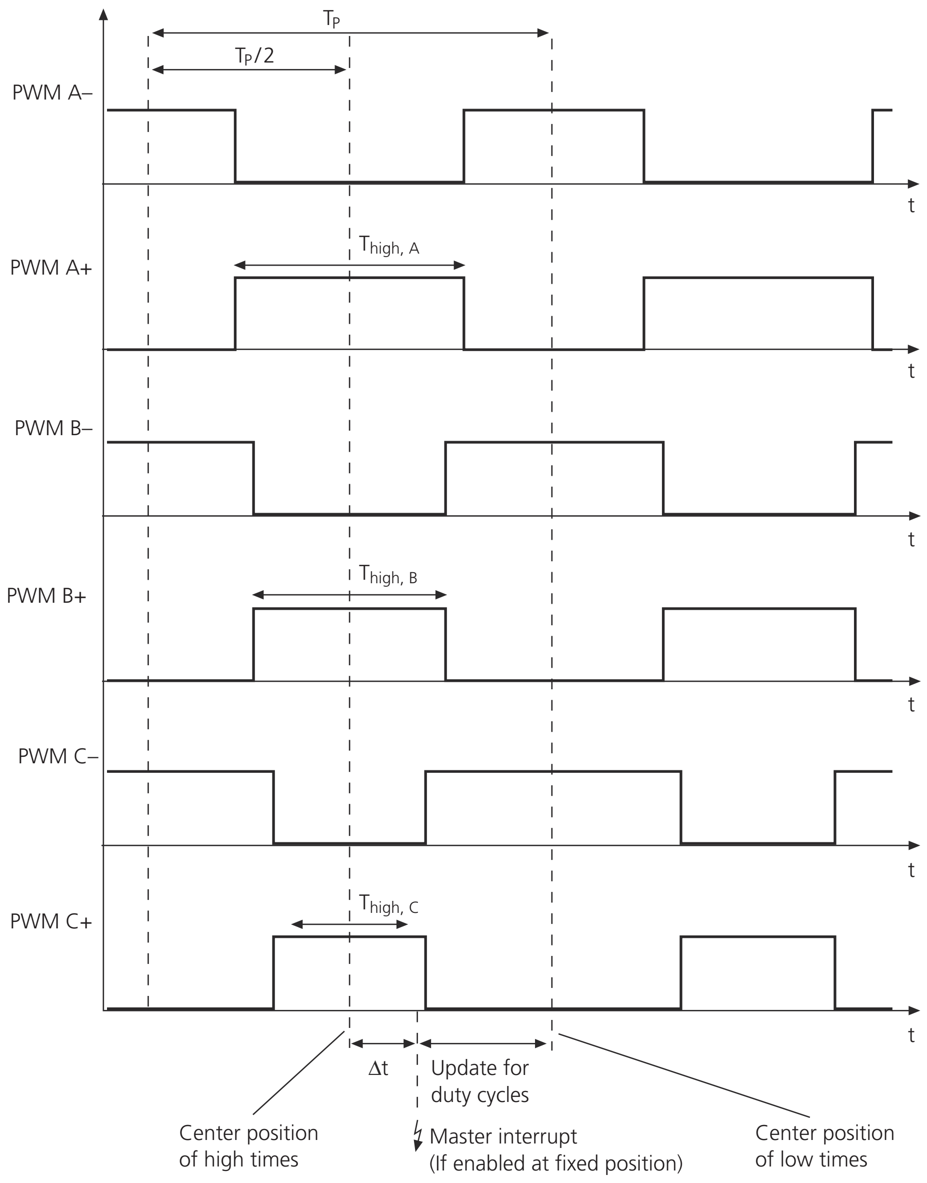
- Run-time adjustable duty cycle (T_{high}/T_p ratio) on each channel
- Variable interrupt position
- Variable dead time value

PWM mode

With this feature you can generate 3-phase PWM signals with the non-inverted signals only (PWM3 mode, upper illustration) or 3-phase PWM signals with the non-inverted and inverted signals (PWM6 mode, lower illustration).


PWM3 mode
T_P
$T_P/2$
$T_{high, A}$
$T_{high, B}$
$T_{high, C}$
PWM A
PWM B
PWM C
t
Δt
Update for duty cycles
Center position of high times
Master interrupt (If enabled at fixed position)
Center position of low times

PWM6 mode



Interrupt positions

The interrupt position can be specified as *fixed* or *variable*. If you select interrupt generation with a fixed position, the interrupt is triggered at the center position of the high level signal. If you select interrupt generation with variable positioning, the interrupt is triggered with a specified time delay ΔI after the center position of the low level signals, refer to the parameter **Interrupt delay** (RapidPro System – I/O Subsystem MPC565 RTI Reference). You can specify the number of PWM periods after which an interrupt is periodically generated. The range is [1 ... 256].

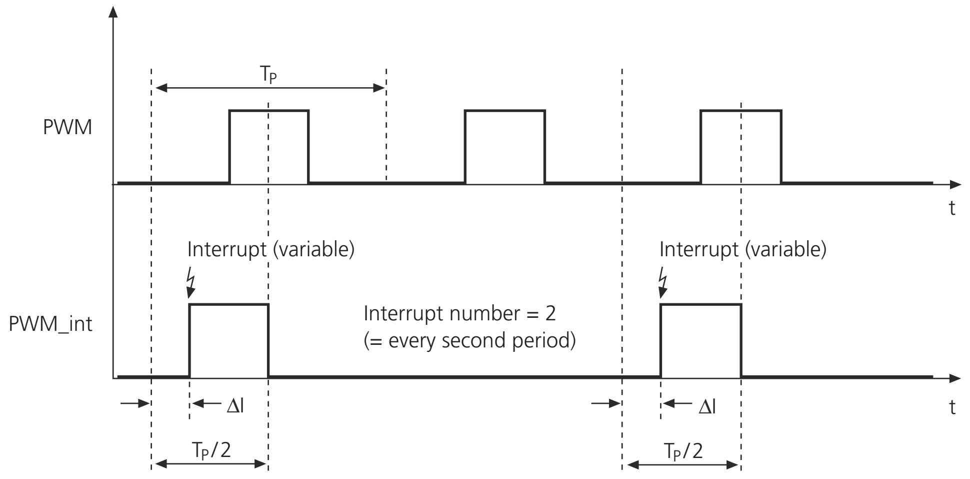
Note

For fixed interrupts, you must consider a constant, system-dependent time delay Δt (PWM3 mode: 4.4 µs, PWM6 mode: 8 µs).

For the use of variable interrupts, an additional PWM signal (see PWM_int in the illustration below) is generated which is specified as follows:

- Rising edge at the same position as the variable interrupt
  By specifying the number of PWM periods after which a variable interrupt is periodically generated, you also specify the number of PWM periods after which the additional PWM signal PWM_int is periodically generated.
- Falling edge at the middle of the period of the center-aligned multi-channel PWM signal

Due to the master-slave communication it might happen that an interrupt reaches the master processor a few microseconds later than specified. The actual delay depends on your application.

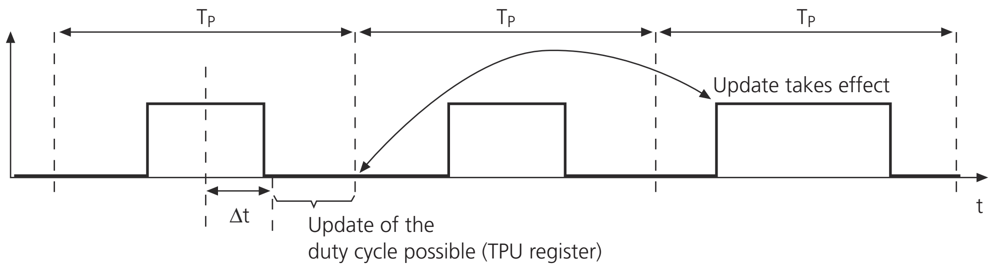


Tip

If the interrupt channel (PWM3: TPU channel 8, PWM6: TPU channel 14) is routed as a digital output channel, you can use PWM_int as an external trigger source for A/D conversion.

Duty cycle update

The duty cycles of the PWM channels used are updated synchronously to the center positions of the low level signals. A duty cycle update can take effect only the period after next.



Note

To guarantee synchronous updating, the periods T_p of the PWM signals must be specified as follows:

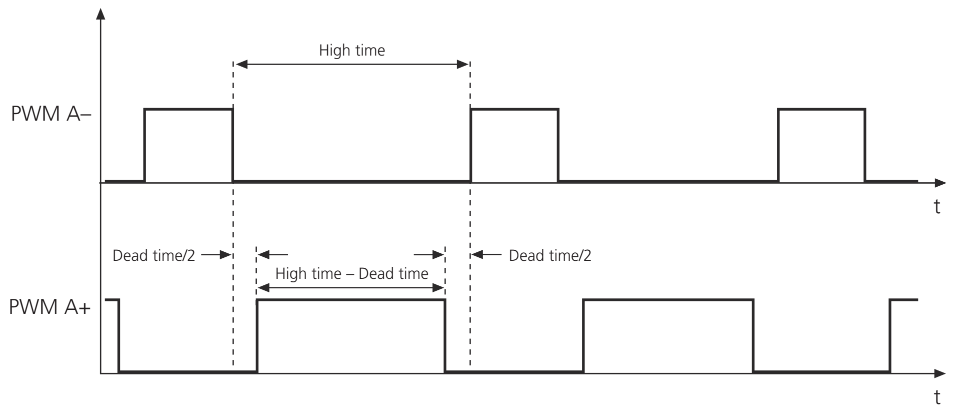
$T_p > 2 \cdot \Delta t$

where Δt is the system-dependent time delay (PWM3 mode: 4.4 µs, PWM6 mode: 8 µs)

Moreover, there must be enough time left (T_p / 2 - Δt) to write all new duty cycle values to the registers of the TPU (depends on the CPU load of the RapidPro system, for example).

Dead time

With the dead time parameter, which is available in PWM6 mode, you can define the timing relationship between each of the 3 PWM output signals and their corresponding inverted output signals, for example, to avoid ripple currents or prevent shoot-through currents on the phase drivers. The high times of the non-inverted output signals are reduced symmetrically by the dead time. The following illustration shows center-aligned multi-channel PWM with a specified dead time.



Note

The maximum dead time value depends on the prescaler. However, it should not be greater than $T_P/2$.

Overview of TPU characteristics

The MCPWM_CA feature is supported only by the TPU. Its characteristics are:

Characteristics	TPU A, B, C
Channel number	Channel assignment must start with the 1st channel of the TPU. Up to 15 channels are successively allocated for the MCPWM_CA feature: ▪ PWM3 mode: 8 successive channels are normally used. 9 successive channels are used if you enable variable interrupt positioning. ▪ PWM6 mode: 14 successive channels are normally used. 15 successive channels are used when you enable variable interrupt positioning.
Update mode	Update of duty cycle synchronously on all 3 or 6 output channels.
PWM mode	▪ PWM3 (3-phase PWM signal generation) ▪ PWM6 (3-phase PWM signal generation with inverted and non-inverted signals)
Interrupt generation	Yes
Interrupt mode	Interrupt can be specified as: ▪ Disabled ▪ Fixed position (in the middle of the high times) ▪ Variable position specified by a time delay, which starts at the middle of the low times
Prescaler range	The prescaling factor depends on the prescaler setup of the specified time counter register.
Period range	Depends on the prescaler and the SC and PS modules used. Refer to Frequency and Dead Time Ranges for Center-Aligned Multi-Channel PWM Signal Generation on page 81.

Channel usage

You must specify the 1st channel of a TPU to configure the MCPWM_CA feature in your model. If channel 1 from TPUs A, B and C are already used by other blocks, it is not possible to implement this feature. The TPU channel 1 is used internally only. Depending on the PWM mode, up to 6 additional channels are allocated automatically for the PWM outputs, and 7 additional channels are allocated for internal use. If you enable interrupt generation with variable interrupt positioning, an additional channel is allocated for interrupt generation.

The channel usage for the PWM3 mode is:

Channel Used	**Meaning**
TPU channel 1	Channel which you must specify in the RTI block or RTLib function. It is not used for output signal generation. It must always be the first channel of the TPU.
TPU channel 2	PWM output channel A (duty 1)
TPU channel 3	Allocated, but not used for output signal
TPU channel 4	PWM output channel B (duty 2)
TPU channel 5	Allocated, but not used for output signal
TPU channel 6	PWM output channel C (duty 3)
TPU channel 7	Allocated, but not used for output signal
TPU channel 8	If variable interrupt positioning is enabled, this channel is used for the additionally generated PWM signal (PWM_int). Otherwise, this channel is allocated, but not used for output signal.
TPU channel 9 (opt.)	Allocated, but not used for output signal, if interrupt generation with variable interrupt positioning is enabled.

The channel usage for the PWM6 mode is:

Channel Used	**Meaning**
TPU channel 1	Channel which you must specify in the RTI block or RTLib function. It is not used for output signal generation. It must always be the first channel of the TPU.
TPU channel 2	PWM output channel A-
TPU channel 3	Allocated, but not used for output signal
TPU channel 4	PWM output channel A+ encl. dead time
TPU channel 5	Allocated, but not used for output signal
TPU channel 6	PWM output channel B-
TPU channel 7	Allocated, but not used for output signal
TPU channel 8	PWM output channel B+ encl. dead time
TPU channel 9	Allocated, but not used for output signal
TPU channel 10	PWM output channel C-
TPU channel 11	Allocated, but not used for output signal
TPU channel 12	PWM output channel C+ encl. dead time
TPU channel 13	Allocated, but not used for output signal

Channel Used	Meaning
TPU channel 14	If variable interrupt positioning is enabled, this channel is used for the additionally generated PWM signal (PWM_int). Otherwise, this channel is allocated, but not used for output signal.
TPU channel 15 (opt.)	Allocated, but not used for output signal, if interrupt generation with variable interrupt positioning is enabled.

Applicable modules

To use the center-aligned multi-channel PWM signal generation feature, your RapidPro system must contain at least one of the following modules:

- SC-DO 8/1
- A PS module, if your RapidPro system contains a Power Unit
- Any other module providing digital output signals

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 🕮).

RTI/RTLib support

You can access this PWM signal generation feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- PWM Signal Generation (PWM) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

Basics

Frequency and Dead Time Ranges for Center-Aligned Multi-Channel PWM Signal Generation

Ranges on the TPU

The frequency and the dead time ranges for the signal generation on a TPU depend on the prescaler setting. The resolution depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54.

Prescaler Value	f_{min}	f_{max}	T_{min}	T_{max}	Dead $Time_{max}$
TCR1					
`TPU_TCR1_PSCK_2`	1709 Hz	100 kHz	10 µs	0.59 ms	4.55 µs
`TPU_TCR1_PSCK_4`	854.5 Hz	50 kHz	20 µs	1.17 ms	9.10 µs
`TPU_TCR1_PSCK_8`	427.3 Hz	25 kHz	40 µs	2.34 ms	18.2 µs
`TPU_TCR1_PSCK_14`	244.2 Hz	14.2 kHz	70.42 µs	4.09 ms	31.8 µs
`TPU_TCR1_PSCK_28`	122.1 Hz	7.14 kHz	140.06 µs	8.19 ms	63.7 µs
`TPU_TCR1_PSCK_42`	81.4 Hz	4.76 kHz	210.08 µs	12.28 ms	95.6 µs
`TPU_TCR1_PSCK_56`	61.1 Hz	3.57 kHz	280.11 µs	16.37 ms	127 µs
`TPU_TCR1_PSCK_84`	40.7 Hz	2.38 kHz	420.17 µs	24.57 ms	191 µs
`TPU_TCR1_PSCK_112`	30.6 Hz	1.78 kHz	561.80 µs	32.68 ms	255 µs
`TPU_TCR1_PSCK_168`	20.4 Hz	1.19 kHz	0.84 ms	49.02 ms	382 µs
`TPU_TCR1_PSCK_224`	15.3 Hz	892 Hz	1.12 ms	65.36 ms	510 µs
`TPU_TCR1_PSCK_336`	10.2 Hz	595 Hz	1.68 ms	98.04 ms	765 µs
`TPU_TCR1_PSCK_448`	7.63 Hz	446 Hz	2.24 ms	131.06 ms	1020 µs
TCR2					
`TPU_TCR2_PSCK_8`	427.3 Hz	25 kHz	40 µs	2.34 ms	18.2 µs
`TPU_TCR2_PSCK_16`	213.7 Hz	12.5 kHz	80 µs	4.68 ms	36.4 µs
`TPU_TCR2_PSCK_24`	142.5 Hz	8.33 kHz	120 µs	7.02 ms	54.6 µs
`TPU_TCR2_PSCK_32`	106.9 Hz	6.25 kHz	160 µs	9.35 ms	72.8 µs
`TPU_TCR2_PSCK_56`	61.1 Hz	3.57 kHz	280.11 µs	16.37 ms	127 µs
`TPU_TCR2_PSCK_64`	53.5 Hz	3.12 kHz	320.51 µs	18.69 ms	145 µs
`TPU_TCR2_PSCK_120`	28.5 Hz	1.66 kHz	600.24 µs	35.09 ms	273 µs

Note

- The described ranges are affected by the TPU workload. If the TPU executes additional functions, the maximum frequency value might not be reached.
- The ranges are theoretical values. In practice, the values depend on the SC and PS modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference 📖.

Related topics

Basics

Timer Setup.......... 53

PWM Signal Measurement (PWM2D)

Objective

The RapidPro Control Unit supports the measurement of duty cycles and frequencies (PWM2D) of up to sixteen PWM signals on each TPU.

Where to go from here

Information in this section

Information in other sections

PWM Signal Measurement (PWM2D) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Information about the RTI blocks concerned with PWM signal measurement.

PWM Signal Measurement (PWM2D) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
Information about the RTLib functions concerned with PWM signal measurement.

Basics of PWM Signal Measurement

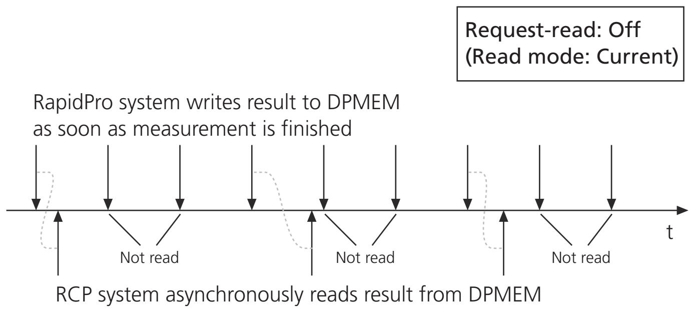
Measurement features

With the PWM2D function, you can measure the frequency of a connected PWM signal and calculate the corresponding duty cycle. The measurement results are written to the dual-port memory (DPMEM) which is used for communication between RapidPro system and the RCP system, for example, MicroAutoBox II or a PHS-bus-based system.

Reading the measurement results

The RapidPro system offers several options for reading the measurement results:

- (Request-read=Off) The RapidPro system writes the results to the DPMEM as soon as a measurement is finished. The RCP system reads the DPMEM asynchronously and retrieves the currently available data.


Request-read: Off
(Read mode: Current)
RapidPro system writes result to DPMEM as soon as measurement is finished
Not read
Not read
Not read
t
RCP system asynchronously reads result from DPMEM

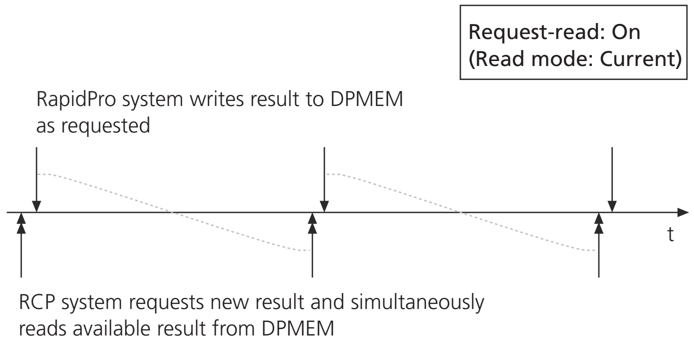
Note

If the frequency of the measured signal is high, the RapidPro system can be overloaded due to writing data to the DPMEM. Additionally, some results are written to the DPMEM but not read by the RCP system.

- (Request-read=On) The RapidPro system writes a new result to the DPMEM only when it has been requested by the RCP system.This is helpful if the frequency of the measured signal is high, as it avoids an unnecessary load on the CPU of the RapidPro system.

  The following two suboptions are possible:

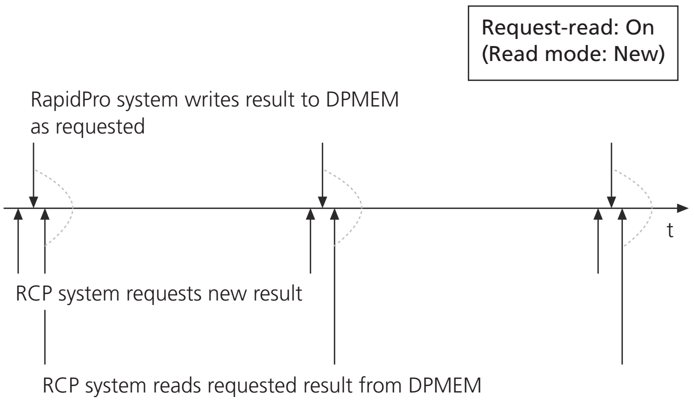
  - (Read mode="Current") The RCP system requests the RapidPro system to write a new result to the DPMEM but simultaneously reads the data that is currently available in the DPMEM.


Request-read: On
(Read mode: Current)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result and simultaneously reads available result from DPMEM

Note

As a rule, the data read by the RCP system represents the last but one sample step.

  - (Read mode="New") The RCP system requests the RapidPro system to write a new result to the DPMEM and waits for the requested data. The RapidPro system writes the requested data to the DPMEM. The RCP system reads the requested data from the DPMEM.


Request-read: On
(Read mode: New)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result
RCP system reads requested result from DPMEM

Note

Model execution on the RCP system slows down if it takes too much time for the RapidPro system to write the requested data to the DPMEM, for example, if the load on the CPU of the RapidPro system is high.

Read status

You can use the status parameter to get information about a result update. If you have read a result, the status is set to *old*. If a new measurement result is available, the status is set to *new*.

Using RTI, you have to enable the status port for access to this information.

Behavior at frequencies outside the range

If the frequency of the measured signal is below the lower limit of the specified frequency range, a value of 0 Hz is returned. The duty cycle then follows the input signal. If you measure a high active PWM signal, the duty cycle is 1 for a high input signal. If you measure a low active PWM signal, the duty cycle is 1 for a low input signal. Whether you measure high active or low active PWM signals depends on the configuration of the SC modules used.

If the frequency of the measured signal is above the upper limit of the specified frequency range, the value of the detected frequency is unpredictable.

Overview of TPU characteristics

The PWM signal measurement feature is supported only by the TPU. Its characteristics are:

Characteristics	**TPU A, B, C**
Channel number	1 … 16
Signal detection	The rising or falling edge can be specified.
Interrupt generation	Yes, if request-read=Off
Prescaler range	The prescaling factor depends on the prescaler setup of the specified time counter register.

Characteristics	TPU A, B, C
Period range	Depends on the prescaler. Refer to Frequency Ranges for PWM Signal Measurement on page 87.
Status	Shows the current status of the PWM measurement: ▪ New: The master has not read the new result value yet ▪ Old: The master has already read the new result value

Applicable modules

To use the PWM signal measurement feature, your RapidPro system must contain at least one of the following modules:

- SC-DI 8/1
- Any other module providing digital input channels

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 📖).

RTI/RTLib support

You can access the PWM signal measurement feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- PWM Signal Measurement (PWM2D) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- PWM Signal Measurement (PWM2D) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

References

Advanced Page (RPCU_PWM2D_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Frequency Ranges for PWM Signal Measurement

Ranges on the TPU

The frequency ranges for the signal measurement on a TPU depend on the prescaler setting. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54.

Prescaler Value	f_{min}	f_{max}
TCR1		
`TPU_TCR1_PSCK_2`	854.5 Hz	700 kHz
`TPU_TCR1_PSCK_4`	427.3 Hz	350 kHz
`TPU_TCR1_PSCK_8`	213.7 Hz	175 kHz
`TPU_TCR1_PSCK_14`	122.1 Hz	100 kHz
`TPU_TCR1_PSCK_28`	61.1 Hz	50.0 kHz
`TPU_TCR1_PSCK_42`	40.7 Hz	33.3 kHz
`TPU_TCR1_PSCK_56`	30.6 Hz	25.0 kHz
`TPU_TCR1_PSCK_84`	20.4 Hz	16.6 kHz
`TPU_TCR1_PSCK_112`	15.3 Hz	12.5 kHz
`TPU_TCR1_PSCK_168`	10.2 Hz	8.33 kHz
`TPU_TCR1_PSCK_224`	7.63 Hz	6.25 kHz
`TPU_TCR1_PSCK_336`	5.09 Hz	4.16 kHz
`TPU_TCR1_PSCK_448`	3.82 Hz	3.12 kHz
TCR2		
`TPU_TCR2_PSCK_8`	213.7 Hz	175 MHz
`TPU_TCR2_PSCK_16`	106.9 Hz	87.5 kHz
`TPU_TCR2_PSCK_24`	71.3 Hz	58.3 kHz
`TPU_TCR2_PSCK_32`	53.3 Hz	43.7 kHz
`TPU_TCR2_PSCK_56`	30.6 Hz	25.0 kHz
`TPU_TCR2_PSCK_64`	26.8 Hz	21.8 kHz
`TPU_TCR2_PSCK_120`	14.3 Hz	11.6 kHz

Note

- The described ranges are affected by the TPU workload. If the TPU executes this function on more than one channel, or executes additional functions, the maximum value might not be reached.
- The ranges are theoretical values. In practice, the values depend on the SC modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference 📖.

Related topics

Basics

Pulse-Width Measurement (PW2D)

Objective

The RapidPro Control Unit provides pulse-width measurement (PW2D).

Where to go from here

Information in this section

Information in other sections

Pulse Width Measurement (PW2D) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Information about the RTI blocks that measure the pulse width of a square-wave signal.

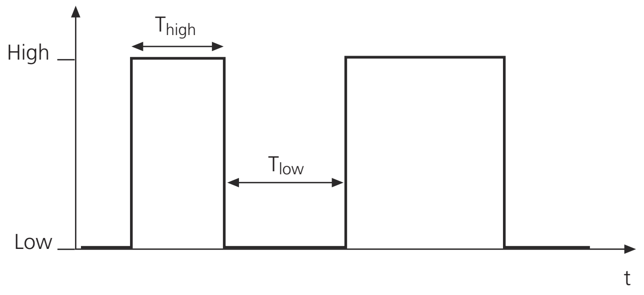
Pulse Width Measurement (PW2D) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Information about the RTLib functions that measure the pulse width of a square-wave signal.

Basics of Pulse-Width Measurement

Definition of "Pulse width"

You can measure the pulse width [s] by applying different edge polarities:

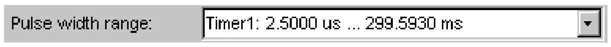
- Edge polarity "Rising edge":
  Pulse width is the time between a rising edge and a falling edge (T_{high})
- Edge polarity "Falling edge":
  Pulse width is the time between a falling edge and a rising edge (T_{low})



Pulse width range

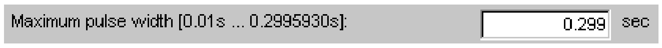
To measure pulse widths, you must specify a pulse width range in advance:

1. Choose a pulse width range (determined by global TPU/MIOS prescaler settings)



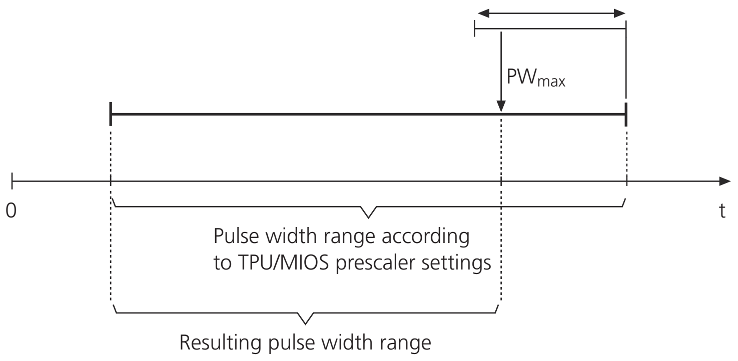
Pulse width range: Timer1: 2.5000 us ... 299.5930 ms

2. Specify an upper pulse width range limit



Maximum pulse width [0.01s ... 0.2995930s]: 0.299 sec

The resulting pulse width range reads as follows:


PWmax
0
t
Pulse width range according to TPU/MIOS prescaler settings
Resulting pulse width range

Averaged pulse widths

If you use the pulse-width measurement functionality of the TPU, you can read the pulse widths from up to 255 periods and calculate their average.

```
T_average = Σ T_1…n / n
n = 1 … 255
```

Note

For low frequencies, the number of measured periods used for calculating the average pulse-width is reduced automatically to prevent a buffer overflow.
If you measure pulse widths on the MIOS, you can read only the pulse width of the current period.

Handling invalid pulse widths

The measurement result is forced to zero, if:

- Measured pulse width is less than the lower pulse width range limit
- Input signal is constantly LOW (edge polarity "Rising edge")
- Input signal is constantly HIGH (edge polarity "Falling edge")

The measurement result is forced to maximum float value (FLT_MAX define is returned by RTLib function), if:

- Measured pulse width is greater than the upper pulse width range limit
- Input signal is constantly HIGH (edge polarity "Rising edge")
- Input signal is constantly LOW (edge polarity "Falling edge")

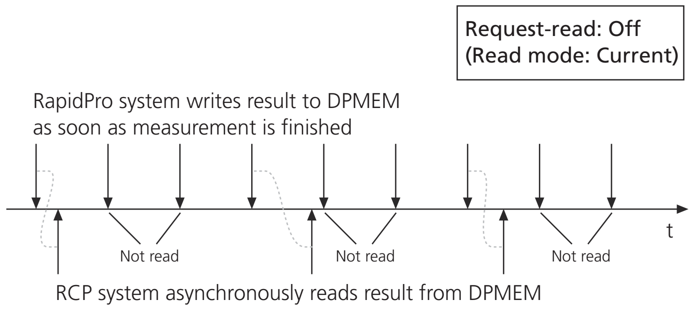
Handling invalid distance between pulses

The measurement result is forced to zero, if the time between consecutive pulses is larger than the upper pulse width range limit.

Reading the measurement results

The RapidPro system offers several options for reading the measurement results:

- (Request-read=Off) The RapidPro system writes the results to the DPMEM as soon as a measurement is finished. The RCP system reads the DPMEM asynchronously and retrieves the currently available data.


Request-read: Off
(Read mode: Current)
RapidPro system writes result to DPMEM as soon as measurement is finished
t
Not read
Not read
Not read
RCP system asynchronously reads result from DPMEM

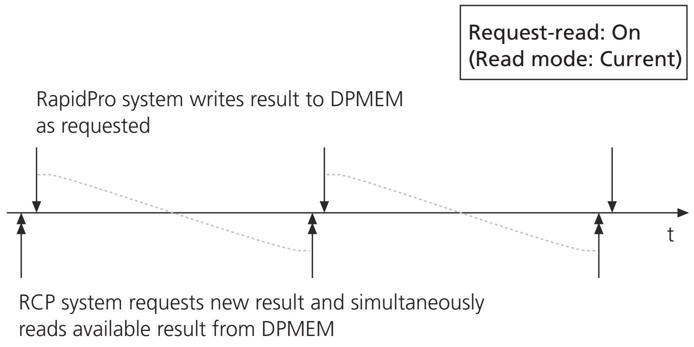
Note

If the frequency of the measured signal is high, the RapidPro system can be overloaded due to writing data to the DPMEM. Additionally, some results are written to the DPMEM but not read by the RCP system.

- (Request-read=On) The RapidPro system writes a new result to the DPMEM only when it has been requested by the RCP system.This is helpful if the frequency of the measured signal is high, as it avoids an unnecessary load on the CPU of the RapidPro system.

  The following two suboptions are possible:

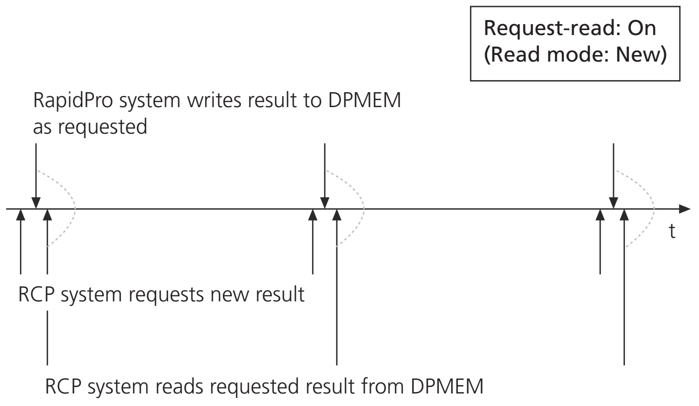
  - (Read mode="Current") The RCP system requests the RapidPro system to write a new result to the DPMEM but simultaneously reads the data that is currently available in the DPMEM.


Request-read: On
(Read mode: Current)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result and simultaneously reads available result from DPMEM

Note

As a rule, the data read by the RCP system represents the last but one sample step.

- (Read mode="New") The RCP system requests the RapidPro system to write a new result to the DPMEM and waits for the requested data. The RapidPro system writes the requested data to the DPMEM. The RCP system reads the requested data from the DPMEM.


Request-read: On
(Read mode: New)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result
RCP system reads requested result from DPMEM

Note

Model execution on the RCP system slows down if it takes too much time for the RapidPro system to write the requested data to the DPMEM, for example, if the load on the CPU of the RapidPro system is high.

Read status

You can use the status parameter to get information about a result update. If you have read a result, the status is set to *old*. If a new measurement result is available, the status is set to *new*.

Using RTI, you have to enable the status port for access to this information.

Overview of TPU/MIOS characteristics

The differences between pulse-width measurement on the TPU and on the MIOS are:

Characteristics	TPU A, B, C	MIOS
Channel number	1 … 16 on each TPU	Allocates always a pair of two channels for measuring two independent signals: 1/2, 3/4, 5/6, 7/8, 9/10

Characteristics	TPU A, B, C	MIOS
Number of periods	1 … 255 Specifies the maximum number of periods to be used for calculating the average pulse width.	1
Interrupt generation	Yes, if request-read=Off	Yes, if request-read=Off
Pulse width range	Depends on the prescaler. Refer to Pulse-Width Ranges for Pulse-Width Measurement on page 94. If the measured pulse width is greater than the upper limit of the pulse width range, the function returns infinity. If the measured pulse width is less than the lower limit of the pulse width range, the function returns zero.	Depends on the prescaler. Refer to Pulse-Width Ranges for Pulse-Width Measurement on page 94. If the measured pulse width is greater than the upper limit of the pulse width range, the function returns infinity. If the measured pulse width is less than the lower limit of the pulse width range, the function returns zero.
Status	Indicates the current status of the pulse-width measurement: ▪ New: The master has not read the new result value yet. ▪ Old: The master has already read the new result value.	

Applicable modules

To use the pulse-width measurement feature, your RapidPro system must contain at least one of the following modules:

- SC-DI 8/1
- Any other module providing digital input channels

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 📖).

RTI/RTLib support

You can access the pulse-width measurement feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- Pulse Width Measurement (PW2D) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- Pulse Width Measurement (PW2D) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC modules and the connected RapidPro Units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

References

Parameters Page (RPCU_PW2D_MIOS_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Parameters Page (RPCU_PW2D_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Pulse-Width Ranges for Pulse-Width Measurement

Ranges on the TPU

The ranges for the pulse-width measurement on a TPU depend on the prescaler setting. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54.

Prescaler Value	T_{min}	T_{max}
TCR1		
TPU_TCR1_PSCK_2	2.50 µs	299 ms
TPU_TCR1_PSCK_4	5.00 µs	599 ms
TPU_TCR1_PSCK_8	10.0 µs	1.19 s
TPU_TCR1_PSCK_14	17.5 µs	2.09 s
TPU_TCR1_PSCK_28	35.0 µs	4.19 s
TPU_TCR1_PSCK_42	52.2 µs	6.29 s
TPU_TCR1_PSCK_56	70.0 µs	8.38 s
TPU_TCR1_PSCK_84	105 µs	12.5 s
TPU_TCR1_PSCK_112	140 µs	16.7 s
TPU_TCR1_PSCK_168	210 µs	25.1 s
TPU_TCR1_PSCK_224	280 µs	33.5 s
TPU_TCR1_PSCK_336	420 µs	50.3 s
TPU_TCR1_PSCK_448	560 µs	67.1 s
TCR2		
TPU_TCR2_PSCK_8	10.0 µs	1.19 s
TPU_TCR2_PSCK_16	20.0 µs	2.39 s
TPU_TCR2_PSCK_24	30.0 µs	3.59 s
TPU_TCR2_PSCK_32	40.0 µs	4.79 s
TPU_TCR2_PSCK_56	70.0 µs	8.38 s
TPU_TCR2_PSCK_64	80.0 µs	9.58 s
TPU_TCR2_PSCK_120	150 µs	17.9 s

Note

- The described ranges are affected by the TPU workload. If the TPU executes the pulse-width measurement on more than one channel, or executes additional functions, the maximum pulse width may not be reached.
- The ranges are theoretical values. In practice, the values depend on the SC modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference 📖.

Ranges on the MIOS

The ranges for the pulse-width measurement on the MIOS depend on the specified prescaler value. The ranges can be calculated using the following equations:

```
T_max = (65534 · P_global · P_channel) / Clock_MPC565 [s]
```

```
T_min = (2 · P_global · P_channel) / Clock_MPC565 [s]
```

```
Clock_MPC565 = 56 MHz
P_global = 2 … 16
P_channel = 0 … 255
```

The following table shows some examples of the minimum and maximum values of the pulse width ranges depending on the prescaler values:

Prescaler Value		T_{min}	T_{max}
Global Prescaler	**Channel Prescaler**		
2	0	0 µs	0 ms
	255	18.21 µs	59.68 ms
4	0	0 µs	0 ms
	255	36.43 µs	1.19 s
8	0	0 µs	0 ms
	255	72.86 µs	2.39 s
16	0	0 µs	0 ms
	255	145.71 µs	4.77 s

Related topics

Basics

Frequency Measurement (F2D)

Objective

The RapidPro Control Unit provides input channels for the frequency measurement of square-wave signals.

Where to go from here

Information in this section

Information in other sections

Frequency Measurement (F2D) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Information about the RTI blocks that measure the frequency of a square-wave signal.

Frequency Measurement (F2D) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
Information about the RTLib functions that measure the frequency of a square-wave signal.

Basics of Frequency Measurement

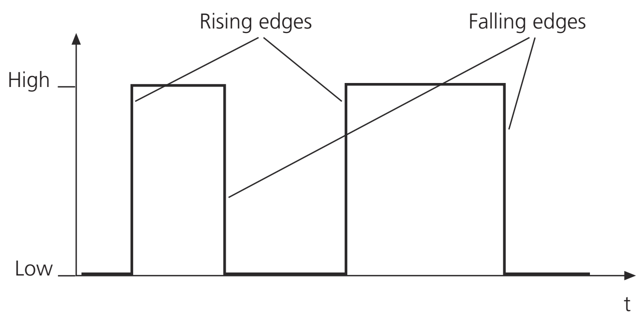
Definition of "Frequency"

You can measure the frequency [Hz] by applying different edge polarities:

- Edge polarity "Rising edge":

  Frequency is the reciprocal value of the time between two consecutive rising edges.
- Edge polarity "Falling edge":

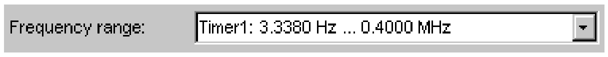
  Frequency is the reciprocal value of the time between two consecutive falling edges.



Frequency range

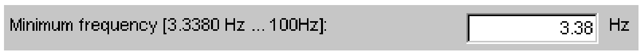
To measure frequencies, you must specify a frequency range in advance:

1. Choose a frequency range (determined by global TPU/MIOS prescaler settings)



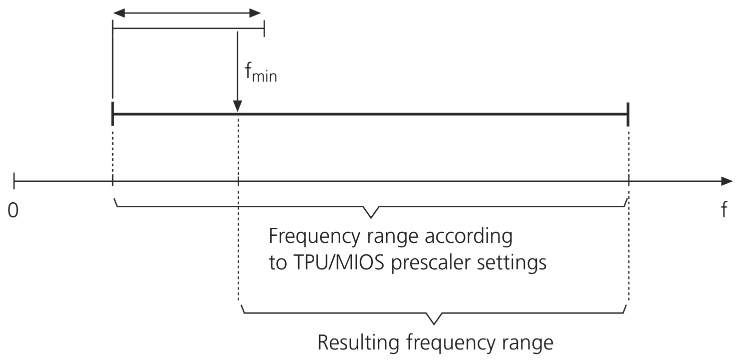
Frequency range: Timer1: 3.3380 Hz ... 0.4000 MHz

2. Specify a lower frequency range limit



Minimum frequency [3.3380 Hz ... 100Hz]: 3.38 Hz

The resulting frequency range reads as follows:


f_{min}
0
f
Frequency range according to TPU/MIOS prescaler settings
Resulting frequency range

Averaged frequencies

If you use the frequency measurement functionality of the TPU, you can read the frequencies from up to 255 periods and calculate their average.

```
f_average = Σ f_1…n / n
n = 1 … 255
```

Note

For low frequencies, the number of measured periods used for calculating the average frequency is reduced automatically to prevent a buffer overflow.
If you use this feature on the MIOS, you can measure only the frequency of the current period.

Handling invalid frequencies

The measurement result is forced to zero, if:

- Measured frequency is less than the lower frequency range limit
- Input signal is constantly LOW
- Input signal is constantly HIGH

The measurement returns unpredictable results, if:

- Measured frequency is greater than the upper frequency range limit

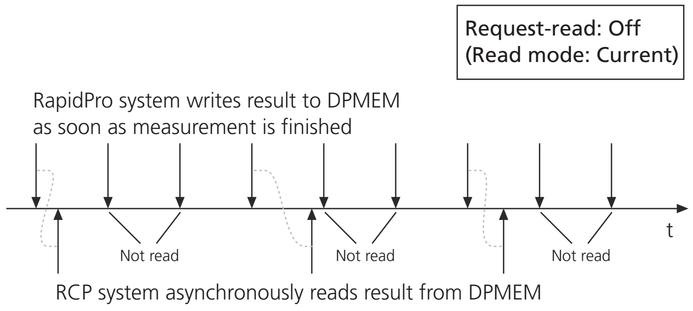
Handling invalid distance between pulses

The measurement result is forced to zero, if the time between consecutive pulses is larger than the reciprocal value of the lower frequency range limit.

Reading the measurement results

The RapidPro system offers several options for reading the measurement results:

- (Request-read=Off) The RapidPro system writes the results to the DPMEM as soon as a measurement is finished. The RCP system reads the DPMEM asynchronously and retrieves the currently available data.


Request-read: Off
(Read mode: Current)
RapidPro system writes result to DPMEM as soon as measurement is finished
t
Not read
Not read
Not read
RCP system asynchronously reads result from DPMEM

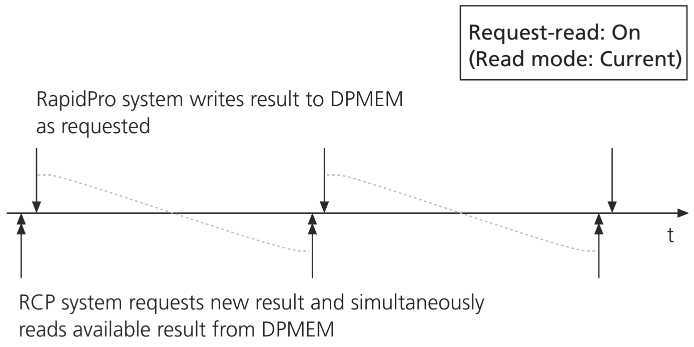
Note

If the frequency of the measured signal is high, the RapidPro system can be overloaded due to writing data to the DPMEM. Additionally, some results are written to the DPMEM but not read by the RCP system.

- (Request-read=On) The RapidPro system writes a new result to the DPMEM only when it has been requested by the RCP system.This is helpful if the frequency of the measured signal is high, as it avoids an unnecessary load on the CPU of the RapidPro system.

  The following two suboptions are possible:

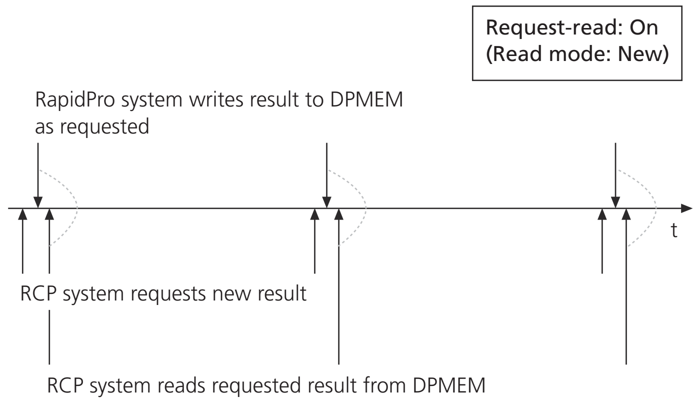
  - (Read mode="Current") The RCP system requests the RapidPro system to write a new result to the DPMEM but simultaneously reads the data that is currently available in the DPMEM.


Request-read: On
(Read mode: Current)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result and simultaneously reads available result from DPMEM

Note

As a rule, the data read by the RCP system represents the last but one sample step.

- (Read mode="New") The RCP system requests the RapidPro system to write a new result to the DPMEM and waits for the requested data. The RapidPro system writes the requested data to the DPMEM. The RCP system reads the requested data from the DPMEM.


Request-read: On
(Read mode: New)
RapidPro system writes result to DPMEM as requested
t
RCP system requests new result
RCP system reads requested result from DPMEM

Note

Model execution on the RCP system slows down if it takes too much time for the RapidPro system to write the requested data to the DPMEM, for example, if the load on the CPU of the RapidPro system is high.

Read status

You can use the status parameter to get information about a result update. If you have read a result, the status is set to *old*. If a new measurement result is available, the status is set to *new*.

Using RTI, you have to enable the status port for access to this information.

Overview of TPU/MIOS characteristics

The differences between frequency measurement on the TPU and on the MIOS are:

Characteristics	TPU A, B, C	MIOS
Channel number	1 … 16 on each TPU	Allocates always a pair of two channels for measuring two independent signals: 1/2, 3/4, 5/6, 7/8, 9/10
Number of periods	1 … 255 Specifies the number of periods to be used for calculating the average frequency.	1
Interrupt generation	Yes, if request-read=Off	Yes, if request-read=Off
Frequency range	Depends on the prescaler. Refer to Frequency Ranges for Frequency Measurement on page 101. If the measured frequency is less than the lower limit of the frequency range, the function returns zero.	Depends on the prescaler. Refer to Frequency Ranges for Frequency Measurement on page 101. If the measured frequency is less than the lower limit of the frequency range, the function returns zero.
Status	Indicates the current status of the frequency measurement: ▪ New: The master has not read the new result value yet ▪ Old: The master has already read the new result value	

Applicable modules

To use the frequency measurement feature, your RapidPro system must contain at least one of the following modules:

- SC-DI 8/1
- Any other module providing digital input channels

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 🕮).

RTI/RTLib support

You can access the frequency measurement feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- Frequency Measurement (F2D) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- Frequency Measurement (F2D) (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC modules and the connected RapidPro Units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

References

Parameters Page (RPCU_F2D_MIOS_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Parameters Page (RPCU_F2D_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Frequency Ranges for Frequency Measurement

Ranges on the TPU

The ranges for the frequency measurement on a TPU depend on the prescaler setting. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54.

Prescaler Value	f_{min}	f_{max}
TCR1		
TPU_TCR1_PSCK_2	3.34 Hz	400 kHz
TPU_TCR1_PSCK_4	1.67 Hz	200 kHz
TPU_TCR1_PSCK_8	835 mHz	100 kHz
TPU_TCR1_PSCK_14	477 mHz	57.1 kHz
TPU_TCR1_PSCK_28	239 mHz	28.5 kHz
TPU_TCR1_PSCK_42	159 mHz	19.0 kHz
TPU_TCR1_PSCK_56	120 mHz	14.2 kHz
TPU_TCR1_PSCK_84	79.5 mHz	9.52 kHz
TPU_TCR1_PSCK_112	59.7 mHz	7.14 kHz
TPU_TCR1_PSCK_168	39.8 mHz	4.76 kHz
TPU_TCR1_PSCK_224	29.9 mHz	3.57 kHz
TPU_TCR1_PSCK_336	19.9 mHz	2.38 kHz
TPU_TCR1_PSCK_448	15.0 mHz	1.78 kHz
TCR2		
TPU_TCR2_PSCK_8	835 mHz	100 kHz
TPU_TCR2_PSCK_16	418 mHz	50 kHz
TPU_TCR2_PSCK_24	279 mHz	33.3 kHz
TPU_TCR2_PSCK_32	209 mHz	25 kHz
TPU_TCR2_PSCK_56	120 mHz	14.2 kHz
TPU_TCR2_PSCK_64	105 mHz	12.5 kHz
TPU_TCR2_PSCK_120	55.7 mHz	6.66 kHz

Note

- The described ranges are affected by the TPU workload. If the TPU executes the frequency measurement on more than one channel, or executes additional functions, the maximum frequency value might not be reached.
- The ranges are theoretical values. In practice, the values depend on the SC modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference .

Ranges on the MIOS

The ranges for the frequency measurement on the MIOS depend on the specified prescaler value. The ranges can be calculated using the following equations:

```
Fmin = Clock_MPC565 / (65535 · P_global · P_channel)  [Hz]
Fmax = Clock_MPC565 / (P_global · P_channel)  [Hz]
Clock_MPC565 = 56 MHz
P_global = 2 … 16
P_channel = 1 … 256
```

The following table shows some examples of the minimum and maximum values of the frequency ranges depending on the prescaler values:

Prescaler Value		f_{min}	f_{max}
Global Prescaler	**Channel Prescaler**		
2	1	427.25 Hz	28 MHz
	256	1.67 Hz	109.37 kHz
4	1	213.63 Hz	14 MHz
	256	0.83 Hz	54.69 kHz
8	1	106.81 Hz	7 MHz
	256	0.42 Hz	27.34 kHz
16	1	53.41 Hz	3.5 MHz
	256	0.21 Hz	13.67 kHz

Related topics

Basics

Incremental Encoder Interface

Objective

The RapidPro Control Unit provides an interface for accessing digital incremental encoder signals.

Where to go from here

Information in this section

Information in other sections

Incremental Encoder Interface (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Information about the RTI block that measures the position and the rotation speed of an incremental encoder.

Incremental Encoder Interface (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Information about the RTLib functions that measure the position and the rotation speed of an incremental encoder.

Basics on Digital Encoder Signals

Encoder signals

Incremental encoders are encoders which provide a series of pulses (increments). These increments are used to analyze position changes. The RapidPro system supports digital incremental encoder with two sensor signals which are shifted by 90°. The shifted signal is used to detect the direction of movement. The outputs of a digital encoder are the square-wave signals PHI0 and PHI90.

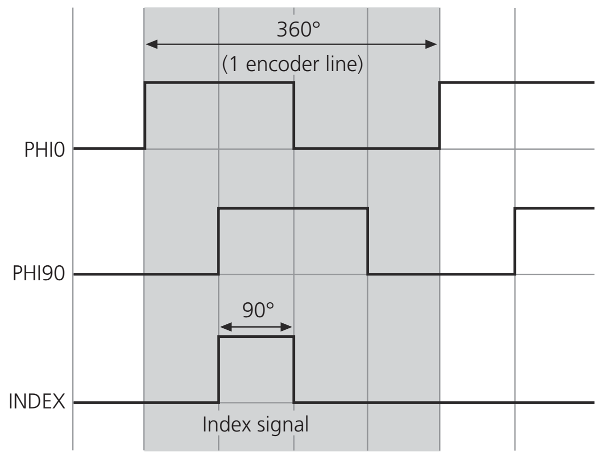
Index signal

Some encoders provide a third signal. This is the index signal IDX. You can use the index signal to detect an initial position. If you use the index signal, you can reset the position counter at the first index signal that occurs or continuously at each index signal.

Shape of the digital sensor output signals

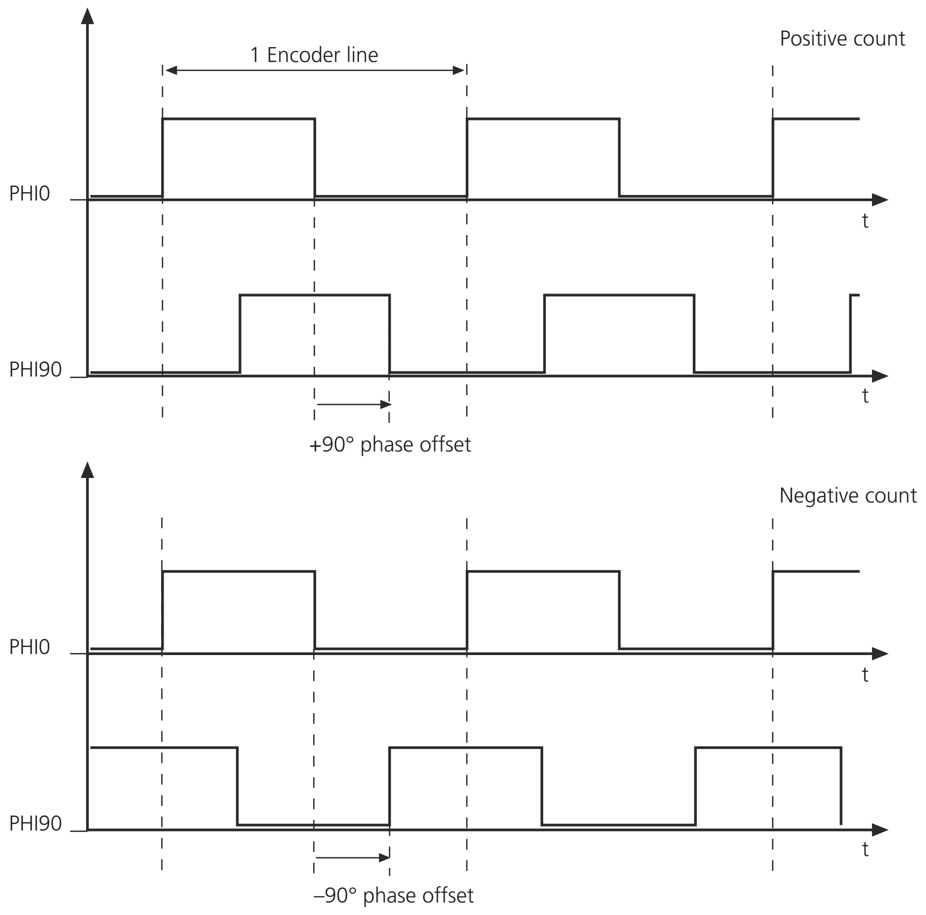
The following illustration shows the shape of the PHI0 and PHI90 digital signals together with the index signal. The gray-shaded area represents one encoder line. The more encoder lines are available for one revolution of the engine to be measured, the more precise is the result of the measurement. To describe the 90° phase shift of the PHI90 signal, one encoder line is represented by 360°. For the position of the index signal, refer to your encoder's data sheet.

Digital signal



Direction of the movement

If two successive edges from the PHI0 and PHI90 signals have the same polarity, the encoder lines are incremented by 1 (positive count). If two successive edges have a different polarity, the encoder lines are decremented by 1 (negative count).



Level of the digital input signals

The relationship between logic level and the required voltage range depends on the SC module which is used for the connected encoder.

Basics on Encoder Line Count

Characteristics of the line count

Note the following points regarding encoder line counting:

- The maximum encoder signal frequency that can be handled by a TPU channel depends on the SC module used.
- Each TPU is equipped with a 16-bit position counter which can be used to count encoder lines on each I/O channel specified for encoder signals. Due to the 4-fold subdivision of each encoder line, the counter allows you to measure up to 2^{14} encoder lines in the range -2^{13} … $+2^{13} - 1$. The count direction depends on the encoder's rotation direction.

- The counter can be reset to a predefined value by the encoder's index signal. You have the following options for counter reset:
  - *Never*
  - *Once* (the counter is reset only after the first index detection)
  - *Continuous* (the counter is reset after each index detection)

4-fold subdivision

The PHI0 and PHI90 signals together provide four edges in one encoder line. Each edge is considered for the analysis of the encoder position, depending on the direction, by incrementing/decrementing the counter by 1. One encoder line is therefore represented by 4 counts in the encoder line counter. RTI blocks and RTLib functions grant access to the quarter positions (position values) of the software counter. This method is also called 4-fold subdivision and is used to increase the accuracy of the measurement.

The following table shows the relationship between the position values and the encoder line counter for one encoder line:

Encoder Line Counter	Position Value
0	0/4 = 0.00
1	1/4 = 0.25
2	2/4 = 0.5
3	3/4 = 0.75
4	4/4 = 1.00

Related topics

Basics

Basics of Incremental Encoder Handling

Objective

The incremental encoder interface provides functions to access digital incremental encoder signals which are connected to the RapidPro hardware. You can read incremental encoder positions and the calculated speed.

Encoder signal type

The incremental encoder interface supports digital incremental encoders that provide single-ended signals. The configuration of the SC module used must suit the connected encoder signals.

Index signal

When you want to determine absolute positions, you can use the index signal to set a position value. The index mode can be set to:

- Not used
  - There is no index signal available or
  - No reaction to a detected index signal
- Reset position value only at the first index pulse

  The position value is reset to the specified initial value at the first occurrence of the index signal. Any subsequent index signal is ignored.
- Reset position value every index pulse

  The position value is reset to the specified initial value at each occurrence of an index signal.

Interrupt generation

If you specified the *Reset position value on first index* or *Reset position on every index* reset modes, you can enable interrupt generation on the index signal. The interrupt can be used to trigger the master. See also Index Found Interrupt on page 356.

Read mode

You can specify two read modes for the encoder channels:

- (Read mode="Current") The RCP system requests the RapidPro system to write a new result to the DPMEM but simultaneously reads the data that is currently available in the DPMEM.
- (Read mode="New") The following actions take place:
  1. The RCP system requests the RapidPro system to write a new result to the DPMEM and waits for the requested data.
  2. The RapidPro system writes the requested data to the DPMEM.
  3. The RCP system reads the requested data from the DPMEM.

  Note

  Model execution on the RCP system slows down if it takes too much time for the RapidPro system to write the requested data to the DPMEM, for example, if the load on the CPU of the RapidPro system is high.

Read status

You can use the status parameter to get information about a result update. If you have read a result, the status is set to *old*. If a new measurement result is available, the status is set to *new*.

Using RTI, you have to enable the status port for access to this information.

Overview of TPU characteristics

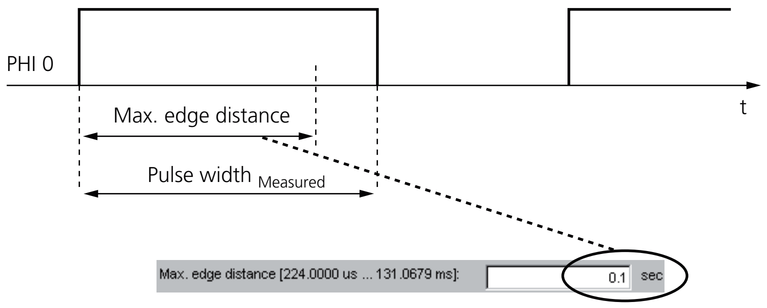
The incremental encoder interface is supported by the time processor unit. Its characteristics are:

Characteristics	TPU A, B, C
Channel number	1 … 14 Always 2 successive channels used
Position range	-8192.00 … +8191.75 (internal 4-fold subdivision)
Start position	-8192.00 … +8191.75
Index position	-8192.00 … +8191.75
Timeout range	If no transition occurs during the specified time, the speed is set to zero (see Conditions forcing encoder speed to zero on page 108 for details).
Pulse width range	Depends on the prescaler. Refer to Encoder Pulse-Width Ranges on page 111.
Prescaler range	The prescaling factor depends on the prescaler setup of the specified time counter register, refer to How to Specify TPU Timers on page 56.
Interrupt generation	Yes Interrupt on index signal
Status	Shows the current status of the incremental encoder interface: ▪ New: The master has not read the new result value yet ▪ Old: The master has already read the new result value

Conditions forcing encoder speed to zero

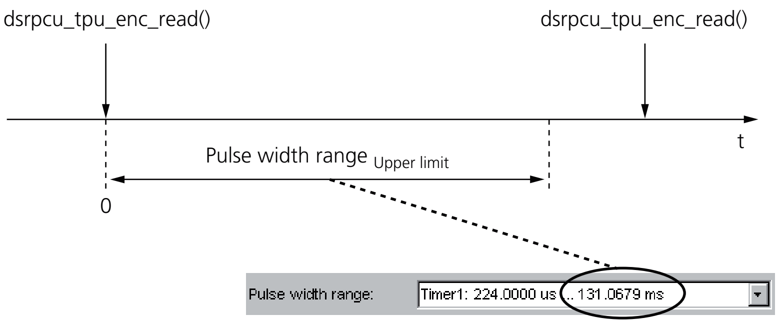
The system sets the speed to zero, if one of the following conditions apply:

- The measured pulse width is larger than the Timeout range (encoder moves too slowly), refer to the Max. edge distance parameter (refer to the RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).



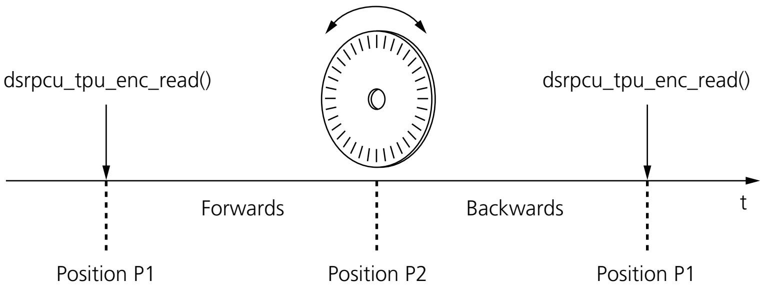
Max. edge distance [224.0000 us ... 131.0679 ms]: 0.1 sec

- The time between consecutive measurements (`dsrpcu_tpu_enc_read()`) is larger than the upper limit of the Pulse width range parameter (refer to the RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).


dsrpcu_tpu_enc_read()
dsrpcu_tpu_enc_read()
t
Pulse width range Upper limit
0
Pulse width range:
Timer1: 224.0000 us ... 131.0679 ms

Solution: Increase the Pulse width range, or decrease the time between consecutive measurements by increasing the sample base rate of the master RTI model.

- Two consecutive measurements (`dsrpcu_tpu_enc_read()`) return the same encoder position. A forwards-backwards-movement of the encoder in between two measurements, for example, is not detected.



Channel usage

You must specify the 1st channel for the incremental encoder. Two further channels are allocated automatically for the second encoder signal and the index pulse, even if the index pulse is not used.

The channel usage for the incremental encoder interface is:

Channel Used	Meaning
1st channel	Used for the index signal if index mode is set to “reset on first index” or “reset on every index” (signal IDX).
2nd channel	1st encoder input channel (signal PHI0)
3rd channel	2nd encoder input channel (signal PHI90)

Note

- With RTI, all 3 channels are always allocated.
- With RTLib, you must specify only the two channels for the encoder input signals, if you do not use the index signal. The first specified channel is then used for the PHI0 signal.
- Some encoder manufacturers use the terms A, B and Z instead of PHI0, PHI90 and Index.
- Encoders with differential RS422 outputs also provide the inverted signals $\overline{\text{PHI0}}$, $\overline{\text{PHI90}}$, and $\overline{\text{IDX}}$. The Control Unit does not support a differential connection to encoders. Encoders with differential outputs can be connected using either the three non-inverted or the three inverted signals. The other three signals must be left unconnected.

Applicable modules

To use the incremental encoder feature, your RapidPro system must contain at least one of the following modules:

- SC-DI 8/1
- Any other module providing digital inputs

For further information, refer to Hardware Overview (RapidPro System Hardware Reference).

RTI/RTLib support

You can access the incremental encoder feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- Incremental Encoder Interface (RapidPro System – I/O Subsystem MPC565 RTI Reference)
- Incremental Encoder Interface (RapidPro System – I/O Subsystem MPC565 RTLib Reference)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

Basics

Encoder Pulse-Width Ranges

Pulse width range

The ranges of the incremental encoder depend on the settings of the prescaler. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54:

Prescaler Value	T_{min}	T_{max}
TCR1		
TPU_TCR1_PSCK_2	2.08 µs	1.17 ms
TPU_TCR1_PSCK_4	4.15 µs	2.34 ms
TPU_TCR1_PSCK_8	8.29 µs	4.68 ms
TPU_TCR1_PSCK_14	14.6 µs	8.19 ms
TPU_TCR1_PSCK_28	29.0 µs	16.3 ms
TPU_TCR1_PSCK_42	43.5 µs	24.5 ms
TPU_TCR1_PSCK_56	58.0 µs	32.7 ms
TPU_TCR1_PSCK_84	87.0 µs	49.1 ms
TPU_TCR1_PSCK_112	116 µs	65.5 ms
TPU_TCR1_PSCK_168	174 µs	98.3 ms
TPU_TCR1_PSCK_224	232 µs	131 ms
TPU_TCR1_PSCK_336	348 µs	196 ms
TPU_TCR1_PSCK_448	464 µs	262 ms
TCR2		
TPU_TCR2_PSCK_8	8.29 µs	4.68 ms
TPU_TCR2_PSCK_16	16.8 µs	9.36 ms
TPU_TCR2_PSCK_24	24.9 µs	14.0 ms
TPU_TCR2_PSCK_32	33.2 µs	18.7 ms
TPU_TCR2_PSCK_56	58.0 µs	32.7 ms
TPU_TCR2_PSCK_64	66.3 µs	37.4 ms
TPU_TCR2_PSCK_120	125.0 µs	70.2 ms

Note

The described ranges are affected by the TPU workload. If the TPU executes additional functions, the maximum encoder pulse width may not be reached.

Related topics

Basics

Timer Setup.. 53

Stepper Motor Control

Objective

The RapidPro Control Unit provides functions to control a stepper motor. You can also specify the acceleration and deceleration.

Where to go from here

Information in this section

Information in other sections

Stepper Motor Control (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Information about the RTI block that controls the actuating signal of a stepper motor.

Stepper Motor (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
Information about the RTLib functions that control the actuating signal of a stepper motor.

Basics of Stepper Motor Control

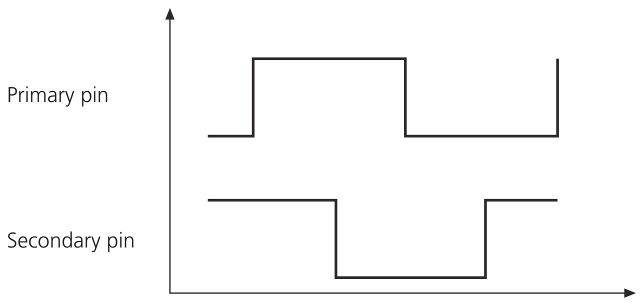
Objective

While the incremental encoder interface can be used for rotation measurement, the stepper motor control can be used for controlling a rotation.

Signal characteristics

Two outputs are available to control the actuating signal of a stepper motor, for example, a throttle valve. The signals are digital, and their levels depend on the SC and PS modules used. The rotating direction of the motor is determined by the sequence of edges of the two signals. You can generate steps within the range -32768 … (32767 - number of acceleration steps). For example, if the position is zero, you can move the motor 32768 steps in negative direction or 32767 steps in positive direction. The resulting number of motor revolutions depends on the stepper motor used.

The signals have the following characteristics:



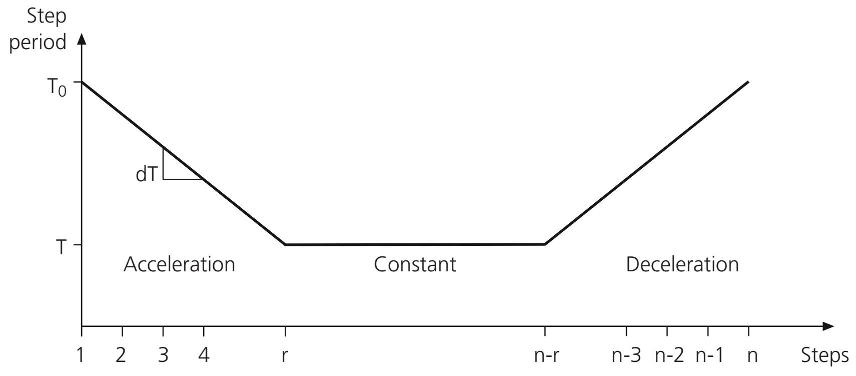
Defining acceleration and deceleration

The number of acceleration and deceleration steps *r – 1* and the step period differences between two consecutive steps *dT* can be used to define the acceleration and deceleration of the stepper motor. The step period *T_0* (step period before acceleration), the step period *T* (step period after acceleration), and the acceleration/deceleration step periods *dT* are limited by a maximum value according to the period ranges. The maximum time interval between two steps is calculated by:

```
T = T_0 – (r – 1) · dT
```

with `(r-1)` : Number of acceleration/deceleration steps

For further information on the period ranges, refer to Step Period Ranges on page 116.



Note

To avoid negative periods, the following inequality has to be observed: *(r – 1) · dT < T_0*.

Overview of TPU characteristics

Stepper motor control is supported only by the TPU. Its characteristics are:

Characteristics	TPU A, B, C
Channel number	1 … 15 Always 2 successive channels used

Characteristics	TPU A, B, C
Acceleration steps	0 … 13
Maximum time interval	Depends on the prescaler. Refer to Step Period Ranges on page 116.
Period difference	Depends on the prescaler. Refer to Step Period Ranges on page 116.
Position range	-32768 ... (32767-Acceleration Steps)
Interrupt generation	No
Prescaler range	The prescaling factor depends on the prescaler setup of the specified time counter register.

Applicable modules

To use the stepper motor control feature, your RapidPro system must contain at least one of the following modules:

- SC-DO 8/1
- A PS module, if your RapidPro system contains a Power Unit
- Any other module providing digital output channels

For further information, refer to Hardware Overview (RapidPro System Hardware Reference 🕮).

RTI/RTLib support

You can access the stepper motor control feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- Stepper Motor Control (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- Stepper Motor (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

For details on the execution times and the corresponding measurement setup, refer to Function Execution Times.

Step Period Ranges

Period range

The ranges of the stepper period depend on the settings of the prescaler. The resolution also depends on the prescaler value and is always the same for all TPU functions which refer to the same timer. For further information, refer to Prescaling on a TPU on page 54:

Prescaler Value	T_{min}	T_{max}
TCR1		
TPU_TCR1_PSCK_2	1.79 µs	1.17 ms
TPU_TCR1_PSCK_4	3.58 µs	2.34 ms
TPU_TCR1_PSCK_8	7.15 µs	4.68 ms
TPU_TCR1_PSCK_14	12.5 µs	8.19 ms
TPU_TCR1_PSCK_28	25.0 µs	16.3 ms
TPU_TCR1_PSCK_42	37.5 µs	24.5 ms
TPU_TCR1_PSCK_56	50.0 µs	32.7 ms
TPU_TCR1_PSCK_84	75.0 µs	49.1 ms
TPU_TCR1_PSCK_112	100 µs	65.5 ms
TPU_TCR1_PSCK_168	150 µs	98.3 ms
TPU_TCR1_PSCK_224	200 µs	131 ms
TPU_TCR1_PSCK_336	300 µs	196 ms
TPU_TCR1_PSCK_448	400 µs	262 ms
TCR2		
TPU_TCR2_PSCK_8	7.15 µs	4.68 ms
TPU_TCR2_PSCK_16	14.3 µs	9.36 ms
TPU_TCR2_PSCK_24	21.5 µs	14.0 ms
TPU_TCR2_PSCK_32	28.6 µs	18.7 ms
TPU_TCR2_PSCK_56	50.0 µs	32.7 ms
TPU_TCR2_PSCK_64	57.2 µs	37.4 ms
TPU_TCR2_PSCK_120	108.0 µs	70.2 ms

Note

The described ranges are affected by the TPU workload. If the TPU executes additional functions, the maximum step period may not be reached.

Related topics

Basics

Timer Setup.. 53

Single Edge Nibble Transmission (SENT)

Objective

SENT is a protocol used between sensors and ECUs to transmit data of high-resolution sensors as an alternative to an analog interface.

You can implement the SENT protocol on the RapidPro system using RTI blocks or RTLib functions.

As of dSPACE Release 7.3, the RapidPro Control Unit RTLib as well as the RapidPro Control Unit RTI blockset support the SAE J2716 SENT2010 standard (contains the transmission/reception of pause pulses).

Note

The RapidPro Control Unit RTI blockset and the RapidPro Control Unit RTLib do not yet provide the functionality to use the RapidPro system as a SENT transmitter.

Where to go from here

Information in this section

Basics on the SENT Protocol

Definition

The SENT (Single Edge Nibble Transmission) protocol is defined in the SAE J2716 standard by the Society of Automotive Engineers (SAE). It is used to transmit data of high-resolution (10-bit or more) sensors as an alternative to an analog interface. The sensor signal is transmitted as a series of pulses.

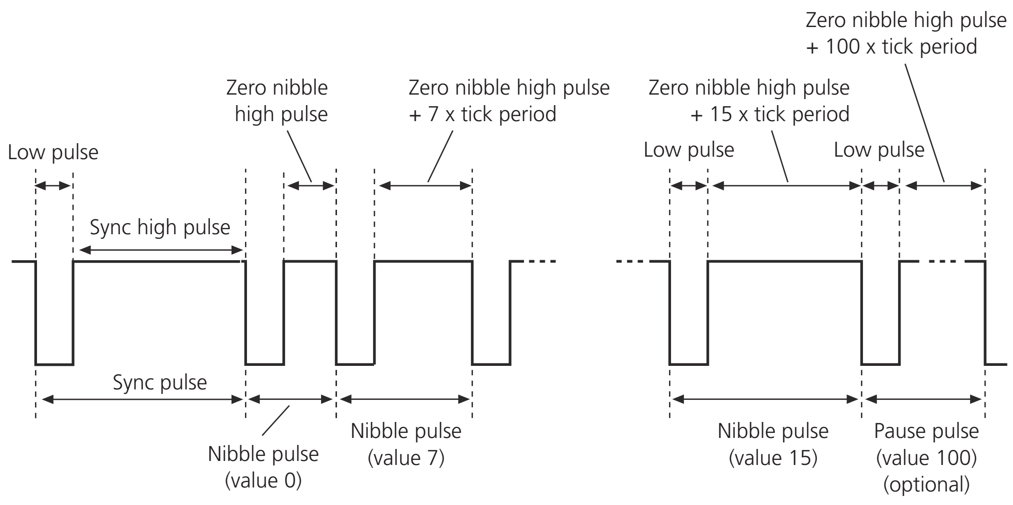
Signal of a SENT message

Every SENT message consists of a sync (synchronization) pulse followed by one or more nibble pulses. For regular SENT specification, the first nibble is a status nibble and the last nibble is a CRC nibble. All other nibbles contain data information.

An optional pause pulse (transmitted at the end of the SENT message) can be used, for example, to create SENT messages with a constant number of tick periods.

The number of nibbles (nibble count) used in every SENT message (including the status and CRC nibble pulses) is always constant for one specific SENT sensor but can vary between different sensors.

The following illustration shows a SENT message with relevant timings and nibble pulses transferring the values 0, 7 and 15 and a pause pulse with a value of 100.


Zero nibble high pulse + 100 x tick period
Zero nibble high pulse
Zero nibble high pulse + 7 x tick period
Zero nibble high pulse + 15 x tick period
Low pulse
Low pulse
Low pulse
Sync high pulse
Sync pulse
Nibble pulse (value 0)
Nibble pulse (value 7)
Nibble pulse (value 15)
Pause pulse (value 100) (optional)

Data information is transmitted via the length of nibble pulses. One SENT nibble pulse consists of one low pulse and one high pulse. The value of a nibble is encoded by the length of a nibble pulse.

High pulse length	= Zero nibble high pulse + Value * Tick period
SENT pulse length	= Low pulse + Zero nibble high pulse + Value * Tick period
Value is the nibble value which is transmitted via SENT pulse. It must be in the range of 0 ... 15.	

Tick period

The lengths of the pulses are based on a clock rate, the tick period. Every SENT pulse is a multiple of this pulse duration. The standard value is 3 µs.

At the beginning of each message, the SENT transmitter sends a sync pulse of a defined number of tick periods. The SENT receiver measures the length of the sync pulse and calculates the current length of the transmitter tick period.

Clock drift A percentage that defines the maximum possible clock drift of a SENT transmitter. The standard value is ±20%.

Pulses of a SENT message

A SENT message provides the following pulses.

SENT pulse A digital signal pulse consisting of a low pulse and a high pulse. There are three different types of SENT pulses, a sync pulse, a nibble pulse and a pause pulse. The meaning of a SENT pulse is detected by its pulse duration and its position in the message. The pulse length is measured between two consecutive falling edges.

Low pulse A pulse that marks the beginning of every SENT pulse. Its length is defined by the number of tick periods. The standard value is 5 tick periods.

High pulse The active part of a SENT pulse. It has two different purposes (zero nibble high pulse, sync high pulse). Its length is defined by the number of tick periods.

Sync high pulse The high part of a SENT synchronization pulse. Its length is defined by the number of tick periods. The standard value is 51 tick periods.

Zero nibble high pulse The high part of a SENT nibble pulse with a value of 0. Its length is defined by the number of tick periods. The standard value is 7 tick periods.

Sync pulse The beginning of every SENT message. It consists of a low pulse followed by a sync high pulse. Its length is defined by the number of tick periods. The pulse length must be longer than the maximum possible duration of a nibble pulse.

Nibble pulse A SENT pulse consisting of a low pulse and a specific high pulse with a minimum length of zero nibble high pulse. The nibble pulse length is defined by the number of tick periods. Every nibble pulse transmits the information of four bits. The pulse length is measured between two consecutive falling edges.

Pause pulse An optional fill pulse consisting of one low pulse and one high pulse with the variable length of n tick periods. It is transmitted at the end of a SENT message after the CRC nibble pulse. The pause pulse can be used, for example, to create SENT messages with a constant number of tick periods.

Implementing SENT Receivers in a Simulink Model Using RTI Blocks

Objective

RTI provides one block for implementing a SENT receiver in a Simulink model. The RPCU_SENT_RX_TPU_BLx block reads the messages stored in the receive FIFO so that their nibbles are available in the Simulink model.

Configuring the channels

To configure channels for receiving SENT messages defined by the SENT specification, you must perform some preparatory steps.

- To realize the electrical interface for a SENT receiver, your RapidPro system must be equipped with suitable RapidPro modules. For details on modules and configuration, refer to Using the SENT Protocol on the RapidPro System on page 131.
- To realize timing features, the RTI software uses the time processor units (TPU) of the RapidPro system. To influence the timing behavior, for example, the tick period, you can specify different timer resolutions for the two available timers for each TPU. These settings are made globally using an RPCU_TIMER_SETUP_TPU_BLx block in your model. For instructions, refer to How to Specify TPU Timers on page 56 .
- For each channel of your RapidPro system which is used as a SENT receiver, drag an RPCU_SENT_RX_TPU_BLx block into your Simulink model and specify the unit and the channel on the Unit page.

Specifying the properties of a SENT signal

The properties of a SENT signal are specified on the Parameters page of the RPCU_SENT_RX_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖) block.

Tick period The expected pulse length of the tick period and the tick period tolerance can be specified. This is the base clock every SENT pulse is generated with. As the RapidPro system measures the actual tick period when a message is received, the block can set a diagnostic flag if the measured length of the tick period is outside the specified tolerance.

Low pulse, zero nibble high pulse, sync high pulse These SENT-specific configurable parameters are defined by a number of tick periods.

Receiving messages and nibbles

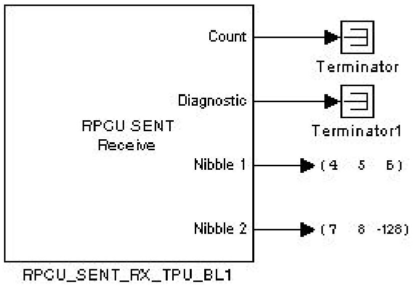
Expected number of messages You have to specify the number of SENT messages which can be stored in the receive FIFO. This number also defines the size of the RPCU_SENT_RX_TPU_BLx block's receive FIFO. If the receive FIFO runs full, the oldest messages stored in the FIFO are lost, because they are overwritten by new input data. In addition an appropriate error flag is set. The number of messages which are currently stored in the receive FIFO is indicated by the Count outport.

Number of nibbles The number of nibbles must include the status nibble and CRC nibble, as they are treated as "normal" nibbles. The RPCU_SENT_RX_TPU_BLx block is given outports for each nibble. The Nibble *n* outport gives a vector with the nibble values of all messages received at the *n* position. The signal connected to the Nibble outports must be a vector whose length is the specified expected number of messages. However, the number of nibbles at a **Nibble** outport matches only the number of messages which are actually received (and indicated by the **Count** outport).

Read mode The block supports different ways of reading messages:

- The block reads all new complete received messages and diagnostic information from the receive FIFO since the last read operation.
- The block reads the most recent message and diagnostic information from the receive FIFO. The receive FIFO is then cleared.

Example The following illustration shows an example of a SENT receiver.



In the example the messages have two nibbles. Three messages are read from the receive FIFO. The first message the nibble values 4 and 7, the second message has 5 and 8. The third message has only one nibble with a value of 6. A value of '-128' is returned for the missing nibble and the error flag in the vector of the Diagnostic outport is set.

Reading received SENT messages

During run time, your application can read the received messages stored in the receive FIFO in two different ways:

- Reading periodically triggered by a timer task

  During run time the timer task calls the **RPCU_SENT_RX_TPU_BLx** block. This is done periodically, for example, every 1.5 ms. To avoid losing received messages, the model cycle should not be longer than the maximum recommended time. For details, refer to Avoiding data loss on page 122.

  The block transfers the new data received since the last read operation to the model. In addition a diagnostic word with diagnostic information is transmitted for each received SENT message. For details, refer to Diagnostics flags (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).
- Reading asynchronously triggered by an interrupt

  To use this method you have to enable interrupt generation by the **Enable Interrupt** parameter of the block.

  In this case, an interrupt is generated on the master system (MicroAutoBox II/DS1007) after a certain number of SENT messages were received, and stored in the receive FIFO. Inside the interrupt service routine, the **RPCU_SENT_RX_TPU_BLx** block can be called to transfer the received message (including diagnostic information) to the model. You can specify the transfer interval via the **Number of messages to trigger interrupt** parameter. For example: If **Number of messages to trigger interrupt** is set to 6, an interrupt is triggered on the master system after every 6th message (which has been received) and these last 6 received SENT messages are transferred to the model with the block call.

Reading pause pulses for diagnostic purposes

Every SENT message can contain a pause pulse transmitted at the end of the SENT message (after the CRC nibble pulse). This optional pause pulse is generated by the transmitter to create SENT messages with a constant number of tick periods.

Pause pulses are recognized and measured by the receiver if you enable the Pause pulse mode block parameter. The values can then be used for diagnostic purposes, for example, as follows: If you specify a non-zero value for the Expected message length block parameter, the receiver uses this expected message length for evaluation. If the message does not match the specified value, diagnostic information is generated.

Note

The RPCU_SENT_RX_TPU_BLx block always interpret the last pulse of a SENT message as a pause pulse, if the Pause pulse mode parameter is enabled.

The possible range of the pause pulse length is automatically calculated from the values specified for SENT block parameters. The range values are displayed on the Parameters page of the block.

Avoiding data loss

The SENT specification requires continuous message reception. To avoid losing received messages due to a full receive FIFO, the RPCU_SENT_RX_TPU_BLx block must be executed periodically.

If the receive FIFO runs full, the oldest messages stored in the FIFO are lost, because they are overwritten by new input data. In addition an appropriate error flag is set. This leads to loss of nibbles or messages. When the RPCU_SENT_RX_TPU_BLx block is executed (for example, in a timer task), the block's receive FIFO is read out and new messages can be received again. The receive FIFO can store as many complete SENT messages as specified by the Expected number of messages (FIFO size) parameter.

Maximum time between two read operations If data is lost, the reason might be that the time between two read operations, i.e., executions of the RPCU_SENT_RX_TPU_BLx block, is too long. The maximum time between two read operations can be calculated as follows:

$T_{\text{Read, max}}$	= Expected number of messages (FIFO size) * $T_{\text{Message, min}}$
$T_{\text{Message, min}}$ (without pause pulse)	= [`SyncHighTics` + `ZeroNibbleHighTics` * `NibbleCount` + `LowTics` * (`NibbleCount` + 1)] * `TicPeriod` * (1 - CD)
$T_{\text{Message, min}}$ (with pause pulse)	= [`SyncHighTics` + `ZeroNibbleHighTics` * (`NibbleCount` + 1) + `LowTics` * (`NibbleCount` + 2)] * `TicPeriod` * (1 - CD)
$T_{\text{Read, max}}$	Maximum time between two consecutive read operations without loss of data.
$T_{\text{Message, min}}$	Minimum possible message duration
`SyncHighTics`	Number of ticks for a synchronization high pulse

`ZeroNibbleHighTics`	Number of ticks for the high part of a SENT nibble pulse with a value of 0.
`NibbleCount`	Number of nibbles included in every SENT message
`LowTics`	Number of ticks for a low pulse
`CD`	Clock drift (tick period tolerance)

The following table shows some values of the maximum time ($T_{Read,max.}$) for different SENT message data if Expected number of messages (FIFO size) = 20 and pause pulse mode is disabled:

TicPeriod	SyncHighTics	ZeroNibbleHighTics	LowTics	NibbleCount	CD	$T_{Read, max.}$
3 µs	51	7	5	6	0.2	6.1 ms
3 µs	51	7	5	10	0.2	8.4 ms
3 µs	51	7	5	6	0.3	5.3 ms
1 µs	51	7	5	6	0.2	2.0 ms

Getting diagnostic information

The RPCU_SENT_RX_TPU_BLx block has three outports which give you status or further information. The outports are optional and must be enabled on the Advanced page.

Tick period The Tick Period outport provides the actual tick period which is measured from the last received valid synchronization pulse.

Diagnostic The Diagnostic outport provides a diagnostic word for each received message. The diagnostic word consists of flags for different message-specific status and diagnostic information. The meanings of the flags are as follows:

Bit (Flag)	Value	Description
0	1	Too many nibbles in message
1	2	Too few nibbles in message
2	4	Nibble value is outside range 0 … 15
3	8	Synchronization pulse too long (out of specified allowed clock drift)
4	16	Synchronization pulse too short (out of specified allowed clock drift)
5	32	Current synchronization pulse differs from the last synchronization pulse by a factor of more than 1/64
6	64	Measured message length differs from expected (fixed) message length.
7	128	The ratio of sync pulse length and complete message length differs by more than 1/64 between the expected value and the current measured value.

For details on the diagnostic flags, refer to Diagnostics flags (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

Error The Error outport provides information on the current read operation. The values have the following meanings:

- 0: No data loss
- 1: Data loss, more messages were received than expected (causing an overflow of the block's receive FIFO)
- 2: Data loss due to overload on the RapidPro I/O subsystem
- 4: Length of a SENT pulse is longer than the maximum pulse length which can be measured by the SENT receiver.

Basics on Receiving SENT Messages Using RTLib Functions

Overview of configurable parameters

Before receiving messages, every SENT receiver must be configured with the following parameters:

Parameter	Type	Description
`NibbleCount`	Initialization	Defines the number of nibbles per SENT message. This number is constant and is set during the initialization of the SENT receiver in the range 1 .. 217.
`LowTics`	Initialization	Defines the length of a low pulse in ticks in the range 1 .. 15. The standard value is 5 tick periods.
`SyncHighTics`	Initialization	Defines the length of the high pulse of a synchronization pulse in ticks in the range 1 .. 255. The standard value is 51 tick periods.
`ZeroNibbleHighTics`	Initialization	Defines the high pulse length of a nibble with a value of 0 in ticks in the range 1 .. 15. The standard value is 7 tick periods.
`TicPeriod`	Initialization	Defines the expected length of a tick period of a SENT pulse in seconds. This is the base clock every SENT pulse is generated with. The standard value is 3 µs.
`ClockDrift`	Initialization	Defines the clock drift that the SENT receiver accepts as valid drift. Synchronization pulses and nibble pulses are detected as valid pulses within this range. Pulses outside this specified range are detected as invalid synchronization pulses or nibble pulses with an invalid value (< 0; >15). When an invalid synchronization pulse is received, the current message is cut and the receiver searches for the next valid synchronization pulse. The diagnostic port reports an invalid synchronization pulse. The range of clock drift is 0 .. 1. The standard value is 0.2 (±20%).
`ReceiveFIFODepth`	Initialization	Specifies the number of SENT messages which can be stored in the receive FIFO in the range 1 … 256.

Parameter	Type	Description
`IntMode`	Initialization	Enables/disables interrupt generation on the master system (MicroAutoBox II/DS1007): If it is enabled, an interrupt is generated on the master system after a SENT message is received.
`IntNumber`	Initialization	Specifies the number of messages which are received and stored in the receive FIFO before an interrupt is triggered on the master system (MicroAutoBox II/DS1007) in the range 1 … 256. For example, if `IntNumber` is set to 6, one interrupt is triggered on the master system after every 6th message that is received.
`PauseMode`	Initialization	Enables/disables pause pulse mode: If the mode is enabled, pause pulses are recognized and measured. The values can be used for diagnostic purposes, for example, as follows: If you enable the `PauseMode` parameter and specify a non-zero value for the `FixedMsgLength` parameter, the receiver uses this expected message length for evaluation. If the message does not match the specified value, diagnostic information is generated.
`FixedMsgLength`	Initialization	Specifies the fixed message length of each message. If you specify a non-zero value, the receiver uses this expected message length for evaluation and can generate diagnostic information. This parameter takes effect only if the `PauseMode` parameter is enabled.

Methods for receiving messages

The SENT receivers support different ways of reading messages. These can be implemented via two different receive functions:

- Read all messages: The `dsrpcu_tpu_sent_rx_receive_all2` function reads all the new messages received completely since the last read operation. If no complete new message is available, nothing is returned.

  If loss of data is detected, the model cycle for reading SENT messages is too long. The model cycle should not be longer than the maximum recommended time. For details, refer to Timing preconditions for receiving messages without data loss on page 127.
- Read most recent message: The `dsrpcu_tpu_sent_rx_receive_most_recent2` function reads the newest complete message. Then the receive FIFO is cleared. If no message was received at all, a message with all nibbles marked as missing nibbles `RPCU_SENT_MISSING_NIBBLE` (-128) is returned.

Reading of received SENT messages

During run time, your application can read the received messages stored in the data buffer in two different ways:

- Reading triggered by a timer task

  During run time the timer task calls a SENT receive function, for example, the `dsrpcu_tpu_sent_rx_receive_all2` function. This is done periodically, for example, every 1.5 ms. To avoid losing received messages, the model cycle should not be longer than the maximum recommended time. For details, refer to Timing preconditions for receiving messages without data loss on page 127.

  The function transfers all new messages received since the last read operation to the model. A diagnostic word with diagnostic information is also transmitted for each received SENT message.

- Reading triggered by an interrupt

  To use this method, you have to enable interrupt generation by the `IntMode` parameter during initialization.

  In this case, an interrupt is generated on the master system (MicroAutoBox II/DS1007) after a SENT message is received and stored in the receive FIFO. In the interrupt service routine, the `dsrpcu_tpu_sent_rx_receive_all2` function (for example) can be called to transfer the received message (including diagnostic information) to the model. You can reduce the transfer interval via the `IntNumber` parameter. For example: If `IntNumber` is set to 6, an interrupt is triggered on the master system after every 6th message (which has been received) and these last 6 received SENT messages are transferred to the model by the `dsrpcu_tpu_sent_rx_receive_all2` function call.

  Note: To avoid data loss, the value of the `IntNumber` should be the same as or less than the value specified by `ReceiveFIFODepth`.

Reading pause pulses for diagnostic purposes

Every SENT message can contain a pause pulse transmitted at the end of the SENT message (after the CRC nibble pulse). This optional pause pulse is generated by the transmitter to create SENT messages with a constant number of tick periods.

Pause pulses are recognized and measured by the receiver if you enable the `PauseMode` parameter. The values can then be used for diagnostic purposes, for example, as follows: If you specify a non-zero value for the `FixedMsgLength` parameter, the receiver uses this expected message length for evaluation. If the message does not match the specified value, diagnostic information is generated.

Note

The `dsrpcu_tpu_sent_rx_receive_all2` and the `dsrpcu_tpu_sent_rx_receive_most_recent2` functions always interpret the last pulse of a SENT message as a pause pulse, if the `PauseMode` parameter is enabled.

The possible range of pause pulse duration [PT_{min} … PT_{max}] (in ticks) follow these formulas:

PT_{min}	= `LowTics` + `ZeroNibbleHighTics`
PT_{max}	= RoundDown (32256 * TR / (TP * (1 + CD)))
TR	TPU timer resolution (TR) = TPRS / CF
TPRS	Prescaler values of the TPU time counter registers: ▪ TCR1: 2 … 448 (13 steps) ▪ TCR2: 8 … 120 (7 steps) The prescaler values can be adjusted via `dsrpcu_tpu_prescaler_set`.
CF	CPU clock frequency: 56 MHz
TP	`TicPeriod`
CD	`ClockDrift`

Timing preconditions for receiving messages without data loss

The SENT specification requires continuous message reception. To avoid losing received messages due to a full receive FIFO, receive functions must be executed periodically. If the receive FIFO runs full, the oldest messages stored in the FIFO are lost, because they are overwritten by new input data. This leads to loss of messages. When a read operation is executed, the receive FIFO is read out and new messages can be received again. The FIFO of a SENT receiver can store as many complete SENT messages as specified by the `ReceiveFIFODepth` parameter.

The maximum time between two read operations to avoid loss of data can be calculated from the following parameters:

$T_{Read,\ max}$	= `ReceiveFIFODepth` * $T_{Message,\ min}$
$T_{Message,\ min}$ (without pause pulse)	= [`SyncHighTics` + `ZeroNibbleHighTics` * `NibbleCount` + `LowTics` * (`NibbleCount` + 1)] * `TicPeriod` * (1 - `ClockDrift`)
$T_{Message,\ min}$ (with pause pulse)	= [`SyncHighTics` + `ZeroNibbleHighTics` * (`NibbleCount` + 1) + `LowTics` * (`NibbleCount` + 2)] * `TicPeriod` * (1 - `ClockDrift`)
$T_{Read,\ max}$	Maximum time between two consecutive read operations without loss of data
$T_{Message,\ min}$	Minimum possible message duration

The following table shows some values of the maximum time ($T_{Read,max.}$) for different SENT message data if `ReceiveFIFODepth` = 20 and pause pulse mode is disabled:

TicPeriod	SyncHighTics	ZeroNibbleHighTics	LowTics	NibbleCount	ClockDrift	$T_{Read,\ max.}$
3 µs	51	7	5	6	0.2	6.1 ms
3 µs	51	7	5	10	0.2	8.4 ms
3 µs	51	7	5	6	0.3	5.3 ms
1 µs	51	7	5	6	0.2	2.0 ms

Generating diagnostic information

The SENT receiver generates a diagnostic information for every received SENT message. The following diagnostic information is evaluated and reported via flags in a single diagnostic word for each received SENT message. The diagnostic information for all received SENT messages since the last read operation is stored in a vector of UInt32. The number of diagnostic words always matches the number of returned messages. The meanings of the flags are as follows:

Bit (Flag)	Value	Description
0	1	Too many nibbles in message.
1	2	Too few nibbles in message.
2	4	Nibble value is outside range 0 … 15.
3	8	Synchronization pulse too long (out of specified allowed clock drift).
4	16	Synchronization pulse too short (out of specified allowed clock drift).
5	32	Current synchronization pulse differs from the last synchronization pulse by a factor of more than 1/64.
6	64	Measured message length differs from expected (fixed) message length.
7	128	The ratio of sync pulse length and complete message length differs by more than 1/64 between the expected value and the current measured value.

Format of message delivered to the model

Messages are delivered to the model as vectors. The delivered data can be interpreted as either a one-dimensional or a 2-dimensional vector, as described below:

One-dimensional vector The format for a one-dimensional vector looks like this:

```
Int8 rx_data[Count * NibbleCount];
```

The driver writes received nibbles to the data buffer one after the other from the first nibble of the first message to the last nibble of the last message. The vector looks like this:

Position in Data Buffer	Message / Nibble
rx_data[0]	Message 1 / Nibble 1
rx_data[1]	Message 1 / Nibble 2
…	…
rx_data[`NibbleCount` - 1] [1]	Message 1 / Nibble `NibbleCount`
rx_data[`NibbleCount`]	Message 2 / Nibble 1

Position in Data Buffer	Message / Nibble
...	...
rx_data[2*`NibbleCount` - 1]	Message 2, Nibble `NibbleCount`
...	...
rx_data[(*M* - 1) * `NibbleCount` + (*N* - 1)] [2]	Message *M*/ Nibble *N*

[1] `NibbleCount` is the number of nibbles per message.
[2] This formula is valid if *M* = 1 … `Count` and *N* = 1 … `NibbleCount`. `Count` is the specified maximum number of messages that are written to the data buffer.

2-dimensional vector The format for a 2-dimensional vector looks like this:

```
Int8 rx_data[Count][NibbleCount];
```

The vector looks like this:

Position in Data Buffer	Message / Nibble
rx_data[0][0]	Message 1 / Nibble 1
rx_data[0][1]	Message 1 / Nibble 2
...	...
rx_data[0][`NibbleCount` - 1] [1]	Message 1 / Nibble `NibbleCount`
rx_data[1][0]	Message 2 / Nibble 1
...	...
rx_data[1][`NibbleCount` - 1]	Message 2 / Nibble `NibbleCount`
...	...
rx_data[(*M* - 1)] [*N* - 1] [2]	Message *M*, nibble *N*

[1] `NibbleCount` is the number of nibbles per message.
[2] This formula is valid if *M* = 1 … `Count` and *N* = 1 … `NibbleCount`. `Count` is the specified maximum number of messages that are written to the data buffer.

Format of pause pulse values delivered to the model

Pause values are delivered to the model as one-dimensional vectors.

The format for a pause value vector looks like this:

```
Int16 rx_pause[n]
```

where n is the number of SENT messages - 1.

If pause pulse mode is enabled, the driver writes received pause pulse values to the data buffer one after the other from the first message to the last message. The vector looks like this:

Position in Data Buffer	Meaning
rx_pause[0]	Pause pulse value of SENT message 1
rx_pause[1]	Pause pulse value of SENT message 2
rx_pause[2]	Pause pulse value of SENT message 3
...	...
rx_pause[n]	Pause pulse value of SENT message n-1

Tick Period Ranges for SENT Messages

Ranges on the TPU

The possible tick period range [TP_{min} … TP_{max}] and the TPU timer resolution (TR) follow these formulas:

TP_{min}	= 5 * TR
TP_{max}	= 32256 * TR / (ST * (1 + CD))
TR	TPU timer resolution (TR) = TPRS / CF
TPRS	Prescaler values of the TPU time counter registers: ▪ TCR1: 2 … 448 (13 steps) ▪ TCR2: 8 … 120 (7 steps) The prescaler values can be adjusted via: ▪ RTI block: RPCU_TIMER_SETUP_TPU_BLx ▪ RTLib function: `dsrpcu_tpu_prescaler_set`
CF	CPU clock frequency: 56 MHz
ST	Low pulse (Number of ticks of a low pulse) + Sync high pulse (Number of ticks of a sync high pulse)
CD	Clock drift (tick period tolerance)

If you use maximum values (*Low pulse* = 15, *Sync high pulse* = 255 and *Clock drift* = 1), the following period ranges [TP_{min} … TP_{max}] are possible:

Prescaler Values		**TP_{min} [µs]**	**TP_{max} [µs]**	**Resolution [ns]**
TCR1	**TCR2**			
2	–	0.18	2.13	35.71
4	–	0.36	4.26	71.43
8	8	0.72	8.53	142.9
14	–	1.25	14.93	250
–	16	1.43	17.06	285.7
–	24	2.15	25.60	428.69
28	–	2.50	29.86	500
–	32	2.86	34.13	571.4
42	–	3.75	44.80	750
56	56	5.0	59.73	1000
–	64	5.72	68.26	1143
84	–	7.5	89.60	1500
112	–	10	119.46	2000
–	120	10.72	128.00	2143
168	–	15	179.20	3000

Prescaler Values		TP_{min} [µs]	TP_{max} [µs]	Resolution [ns]
TCR1	TCR2			
224	–	20	238.93	4000
336	–	30	358.40	6000
448	–	40	477.86	8000

Note

- The ranges are affected by the RapidPro I/O subsystem's workload. If the I/O subsystem executes this function on more than one channel, or executes additional functions, the minimum tick period value might not be reached.
- The ranges are theoretical values. In practice, the values depend on the SC modules used. For further information, refer to the module data sheet in the RapidPro System Hardware Reference.

Using the SENT Protocol on the RapidPro System

Software support

You can implement the SENT protocol on a RapidPro system to use it as a SENT receiver via RTI blocks or RTLib functions.

Note

The RTI blockset and the RTLib functions do not yet provide the functionality to use the RapidPro system as a SENT transmitter.

Required RapidPro hardware

To realize the electrical interface for a SENT receiver, your RapidPro system must be equipped at least with the following RapidPro modules:

- SC-DI 8/1 module (DS1642)
- SC-CCDI 6/1 module (DS1637)
- Any other module providing suitable digital input channels

  If you use such a module, you have to ensure that its electrical interface is compatible with the specification in the SENT standard.

To connect the SENT signal, you need only one digital input channel of a suitable module like the DS1642 or DS1637. This channel must be mapped to the first of two required TPU channels (to the channel with the lowest number). The second TPU channel is used only internally and can be connected to an arbitrary signal source (if it is routed at all).

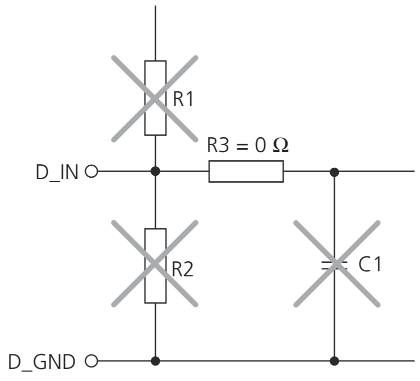
The I/O mapping between the specified signals and the I/O connector pins depends on the installed modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed in ConfigurationDesk for RapidPro. For further information, refer to Signal Mapping to I/O Pins on page 20.

Software configuration You have to configure each channel of a SC-DI 8/1 or SC-CCDI 6/1 module which is used as a SENT receiver in ConfigurationDesk for RapidPro as follows:

Parameter	Setting
Upper threshold	+ 2.0 V
Lower threshold	+ 0.8 V
Signal polarity	Non-inverted

The settings are the factory default settings.

Hardware configuration The user-defined network of each channel of an SC-DI 8/1 or SC-CCDI 6/1 module which is used as a SENT receiver should be equipped with the following components:

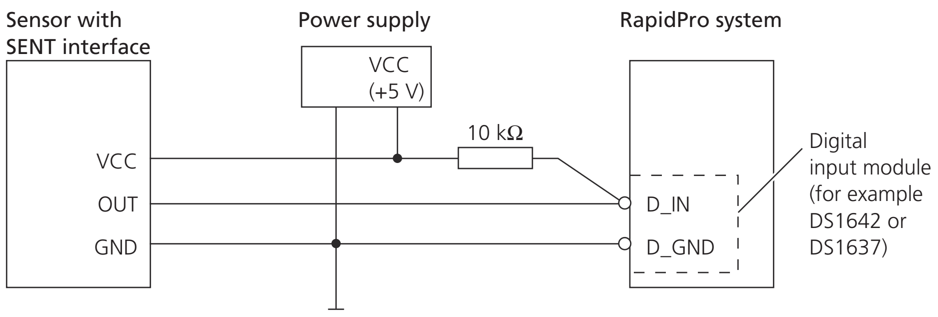


Component	Value
R1	Not equipped
R2	Not equipped
R3	0 Ω
C1	Not equipped

For more hardware details on the modules, for example, the location of solder pads, refer to:

- SC-DI 8/1 Module (RapidPro System Hardware Reference 🕮)
- SC-CCDI 6/1 Module (RapidPro System Hardware Reference 🕮)

Connecting inputs The electrical interface of a SENT receiver is defined in the SENT standard. To adapt the channels which are used as a SENT receiver to this specification, you have to connect the inputs as shown in the illustration below:


Sensor with SENT interface
Power supply
RapidPro system
VCC (+5 V)
10 kΩ
VCC
OUT
GND
D_IN
D_GND
Digital input module (for example DS1642 or DS1637)

- Connect an external pull-up resistor with a resistance value of 10 kΩ from each input pin (D_IN) of the module to the +5 V power supply of the SENT sensor.
- Connect ground of the +5V power supply to a ground pin (D_GND) of the module.

Notes on the CPU workload

To get an idea of the CPU workload of the RapidPro system when receiving messages, the following table shows the workload of a typical SENT message:

Tick Period	Tics for Sync High Pulse	Tics for Zero Nibble High Pulse	Tics for Low Pulse	Nibble Count	CPU Workload
3 µs	51	7	5	4	16 %

When estimating the CPU workload, you should also note the following dependencies:

- If you use several SENT receivers in parallel in your RapidPro system, their workloads must be added.
- A lower value for the nibble count increases the CPU workload. A SENT message with nibble count = 1 generates a workload of approx. 22 %.
- The tick period influences the workload as follows (if the number of ticks for sync high pulse, zero nibble high pulse, low pulse and nibble count do not change):
  - If the tick period doubles, the CPU workload approx. halves.
  - If the tick period halves, the CPU workload approx. doubles.

Reducing CPU workload You can reduce the CPU workload, for example, to avoid data loss, by modifying your Simulink model as follows:

- Reduce the number of SENT receivers.
- Increase the model cycle time.
- Reduce the number of RTI blocks from the RapidPro Control Unit RTI blockset which can read input signals, such as the RPCU_BIT_IN_BLx, RPCU_ADC_BLx or the RPCU_PWM2D_TPU_BLx block.

Engine Control

Objective

The blockset of the RapidPro Control Unit lets you conveniently implement Simulink models to control the behavior of internal combustion engines.

Where to go from here

Information in this section

Information in other sections

Engine Control (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Gives information about the RTI blocks concerned with engine control features.

Engine Control (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
Gives information about the RTLib functions concerned with engine control features.

Introduction to the Engine Control Features

Where to go from here

Information in this section

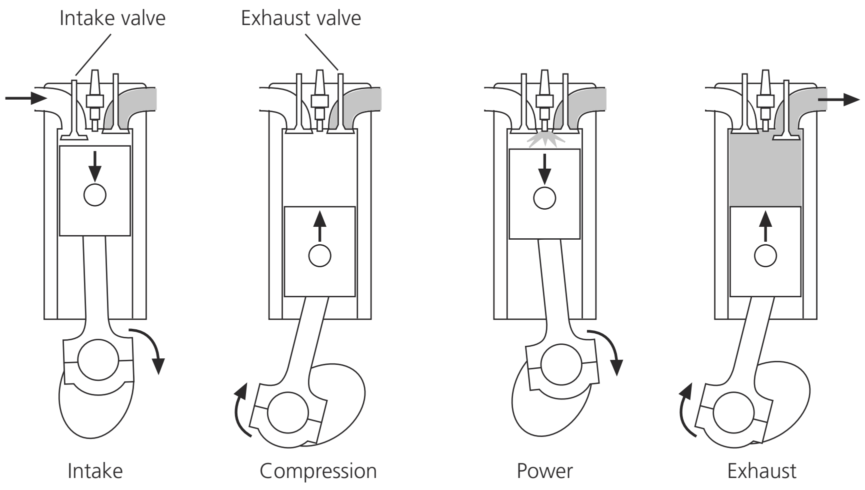
Operation Principle of Four-Stroke Piston Engines

Objective

Outlines the basic functionality and terminology of four-stroke piston engines.

Four-stroke cycle

Today, internal combustion engines in cars, trucks, motorcycles, construction machinery and many others, most commonly use a four-stroke cycle.



The four strokes make up one engine cycle and are as follows:

1. Intake:

   The piston descends from the top to the bottom of the cylinder, reducing the pressure inside it. A mixture of fuel and air is forced into the cylinder. The intake valves then close.

2. Compression:

   With both intake and exhaust valves closed, the piston returns to the top of the cylinder, compressing the fuel-air mixture.

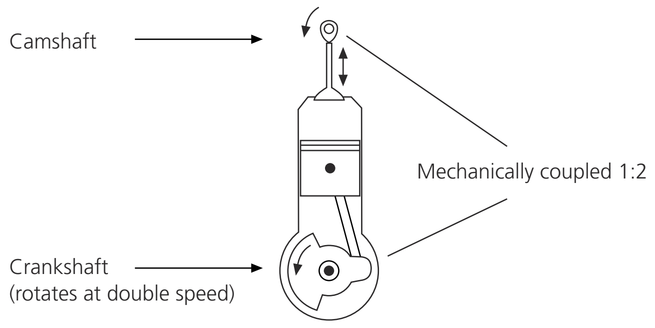
3. Power:

   While the piston is at or close to top dead center, the compressed air–fuel mixture is ignited, usually by a spark plug (for a gasoline or spark ignition engine) or by the heat and pressure of compression (for a diesel cycle or compression ignition engine). The massive pressure resulting from the combustion of the compressed fuel-air mixture drives the piston back down toward bottom dead center with tremendous force.
4. Exhaust:

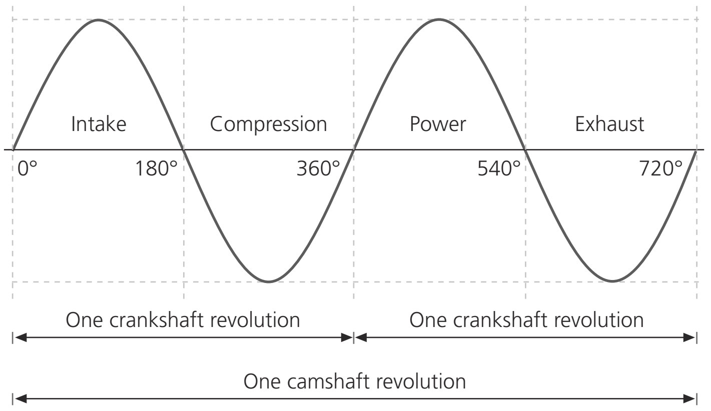
   The piston once again returns to top dead center while the exhaust valve is open. This action evacuates the products of combustion from the cylinder by pushing the spent fuel-air mixture through the exhaust valves.

Crankshaft and camshaft

The reciprocating linear piston motion is translated into crankshaft rotation. The crankshaft transmits the engine power to the wheels via the gearbox. Commonly, one or even multiple camshafts control the intake and exhaust valves. They are mechanically coupled to the crankshaft. The four-stroke operation principle requires that the crankshaft rotates at double speed.

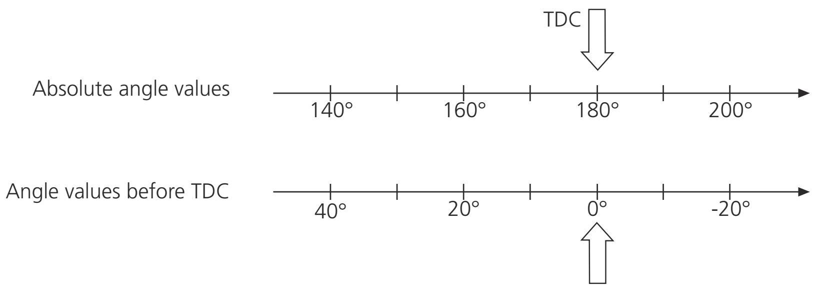


One engine cycle of a four-stroke piston engine corresponds to two crankshaft revolutions and one camshaft revolution. The angular position of the crankshaft corresponds to the position of the piston. Thus, if you know the position of the crankshaft, you can derive the position of the piston. However, you do not know which stroke it is (compression, exhaust, etc.) unless you also know the angular position of the camshaft.



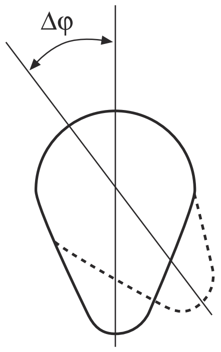
Top dead center

Top dead center (TDC) is the upper reversal point of a piston after compression and exhaust, and is represented by an angle value in the range 0° … 720°. TDC is a cylinder-specific attribute, and each cylinder has its own TDC. TDC is the datum point from which engine timing measurements are made. For example, injection and ignition timing is normally specified relative to the TDC as degrees before top dead center (BTDC). Positions before TDC have positive angle values, positions after TDC have negative ones.



Camshaft phase shift

A camshaft phase shift is commonly a function of speed and load, and aims to save fuel and increase power. It means an angular displacement of a camshaft relative to the coupled crankshaft during operation, triggered by an external mechanism. The graphic below illustrates the angular displacement Δφ of a cam when the camshaft is shifted.



Impact of the Crankshaft Wheel Specification

Objective

Outlines the impact of the two available specification approaches on the engine control features.

Crankshaft wheel specification

You can specify a crankshaft wheel as follows:

- Via parameters for older RapidPro software versions, or
- Via a wavetable, which is a new approach

The wavetable-based specification improves the engine control functionality.

Benefit of the wavetable approach

Specifying a crank wheel via wavetable is advantageous because:

- The gaps and teeth of a crankshaft wheel can have arbitrary length.
- Asymmetric crankshaft wheels can be used.
- No need to specify a camshaft evaluation segment.
- Synchronization can be achieved faster.
- Speed measurement is more precise if speed changes abruptly, for example, at a gap.
- Speed measurement results can be smoothed.
- Reverse crankshaft rotations can be handled.

Tip

Your RapidPro software provides a set of wavetables that relate to commonly used crank wheel designs. In addition, you can use an M script to generate your own wavetables.

For details, refer to Basics on Crankshaft Wheel Specification on page 201 and Basics on the Synchronization of the Crankshaft and Camshaft Signals (Wavetable-Based) on page 150.

System requirements for the wavetable approach

You must use a RapidPro Control Unit with MPC565 not older than board revision DS1601-12 in the wavetable specification approach.

Limitation of the wavetable approach

You cannot use/specify crank wheels without a gap in the wavetable specification approach.

Input Signals Required for Engine Control

Objective

The RapidPro engine control features require specific input signals. The section provides basic information on these signals, which you should be familiar with when you set up the measurement of the crankshaft and camshaft signals.

Note

Your RapidPro system uses crankshaft and camshaft signals, but is not involved in generating them.

Where to go from here

Information in this section

Information in other sections

Basics on Required Crankshaft/Camshaft Signals

Objective

Engine control requires a crankshaft signal and a camshaft signal which describe the motion of these engine parts.

Where to go from here

Information in this section

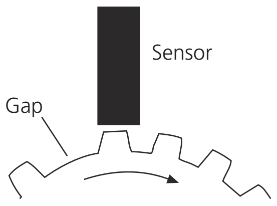
Basics on Crankshaft Sensor Signals

Objective

The crankshaft signal describes the motion of the crankshaft and is required for engine control. This section provides basic information about generating the crankshaft signal using a crankshaft wheel.

Crankshaft sensor

An optical or magnetic sensor scans the radial surface of the rotating crankshaft wheel. The crankshaft wheel is an even toothed wheel flanged to the crankshaft. A gap is a place where teeth are missing.

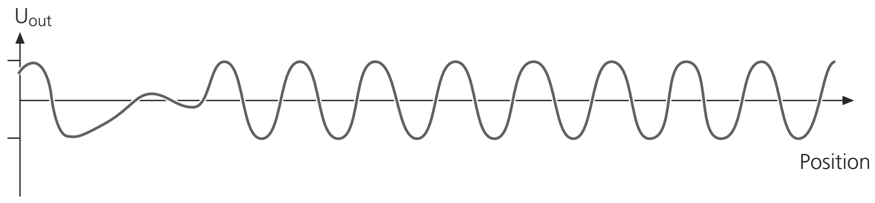


The crankshaft sensor generates a signal that describes the motion of the crankshaft. The crankshaft signal is particularly used for speed measurement and, together with the camshaft signal, for angle measurement, which is required for the generation of injection and ignition pulses.

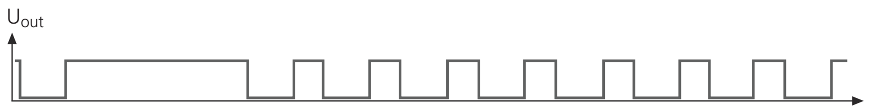
Sensor type

Crankshaft signals are either passive or active, depending on the type of sensor you use. The amplitude of the signal generated also depends on the sensor.

Passive sensors Generate an analog signal, for example:



Active sensors Generate a digital signal (e.g., TTL signal), for example:



Intelligent crank sensor with direction detection

To detect reverse crankshaft rotation, a more sophisticated type of crankshaft sensor is needed. Reverse crankshaft rotation happens, for example, in the following situations:

- Pressure in cylinders due to compression is reduced after engine shutdown.
- The engine is restarted according to combustion, as in modern stop-start automatics.

So-called *reverse crankshaft sensors* can get exact information about the crankshaft position, even on reverse rotation of the crankshaft. Reverse crankshaft sensors generate pulse signals (active sensor). The temporal pulse length of these signals depends on the current rotation direction (e.g., forward: tf=45 µs, reverse: tr=90 µs).

Preconditions To handle reverse crankshaft rotations, the crank wheel must be specified via a wavetable and the reverse crank operation mode must be activated, refer to Parameters Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Related topics

Basics

Basics on Camshaft Sensor Signals........145
Basics on the Synchronization of the Crankshaft and Camshaft Signals (Wavetable-Based)........150
Possible Crankshaft Wheels........144
Processing the Camshaft Signal........148

HowTos

How to Set Up Crankshaft Signal Processing........205

References

Crankshaft Signal Reference Data........271

Possible Crankshaft Wheels

Objective

The engine control features of the RapidPro Control Unit support symmetric and asymmetric crankshaft wheels.

Symmetric crankshaft wheels

A crankshaft wheel is called symmetric if all the following conditions are met:
- There are the same number of teeth between all the gaps (segments).
- All the gaps comprise the same number of missing teeth.
- All the teeth are the same length.

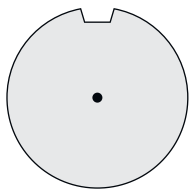
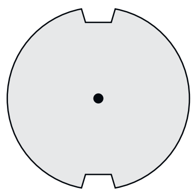
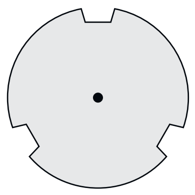
Note

Wheels that are named with respect to the crankshaft wheel parameters, for example, 60 – 2 wheel, are symmetric wheels.

Segments A segment comprises the following angle range:

Segment = 360° / number of gaps

Example These are some examples of symmetric crankshaft wheels (schematic illustration):

1 gap, segment = 360°	2 gaps, segment = 180°	3 gaps, segment = 120°
		

Asymmetric crankshaft wheels

A crankshaft wheel is asymmetric if one or more of the following conditions are met:
- The number of teeth between the gaps differs.
- The number of missing teeth in the gaps differs.
- The length of the teeth differs.

Note

The specification of asymmetric crankshaft wheels is only possible by referencing a wheel wavetable, refer to Basics on Crankshaft Wheel Specification on page 201.

Related topics

Basics

References

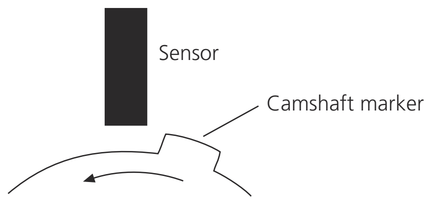
Basics on Camshaft Sensor Signals

Objective

The camshaft signal describes the motion of the camshaft. It is used in combination with the crankshaft signal to identify the exact engine position and to measure any camshaft phase shift.

Camshaft sensor

An optical or magnetic sensor scans the radial surface of the rotating camshaft wheel and detects the edges of the marker(s). The camshaft wheel is a smooth wheel with one or more markers and is flanged to the camshaft.



Camshaft markers

One or more markers are located on the surface of the camshaft wheel. A marker is specified by a start angle and an end angle. All angle values of the camshaft marker specification relate to the crankshaft signal. The possible camshaft angle range is (0° … 719.9°), as one camshaft revolution represents one engine cycle and corresponds to two crankshaft revolutions.

Active or passive camshaft signal

The camshaft sensor generates either a passive or an active signal, depending on the type of sensor you use. The amplitude of the signal generated also depends on the sensor. Refer to Basics on Crankshaft Sensor Signals on page 142.

Number of camshaft wheels

The maximum number of camshaft wheels that can be used depends on the implementation:

- Up to four camshaft wheels can be used in a Simulink model (RTI blocks),
- Up to 16 camshaft wheels can be used with RTLib functions

Note

If the crankshaft wheel does not have a gap, only up to four camshaft wheels can be used for synchronizing angle measurement, even if the model is implemented by RTLib functions.

Related topics

Basics

HowTos

References

Crankshaft/Camshaft Signal Processing

Objective

The crankshaft and camshaft signals are evaluated so that the engine speed and the engine position can be measured.

Where to go from here

Information in this section

Processing the Crankshaft Signal

Objective

This section provides basic information on processing the crankshaft signal so that it can be used by the RapidPro engine control features.

Signal conditioning via SC modules

The incoming crankshaft signal must be connected to an SC module. The SC module must be mounted on the RapidPro Control Unit, not on a RapidPro SC Unit mounted on top of the RapidPro Control Unit. Both low and high active crankshaft signals can be processed. It is not necessary to invert them externally.

Note

An external inversion of the crankshaft signal, for example, by ConfigurationDesk for RapidPro, is even not allowed.

Tip

To change the polarity of the crankshaft signal use the **Edge polarity** parameter of the RPCU_CRANK_SETUP_TPU_BLx block.

SC modules let you scale the crankshaft signal according to your needs. Refer to Configuring RapidPro Hardware (ConfigurationDesk for RapidPro - Guide 🕮). They can also perform noise filtering on an incoming crankshaft signal (if implemented).

You have to use different SC modules, depending on the type of the crankshaft signal. For detailed information on the SC modules, refer to RapidPro System Hardware Reference 🕮.

Generation of equidistant 0.1° pulses

From the SC module, the crankshaft signal is routed to the angular computation unit (ACU). The ACU is a part of the I/O PLD. The ACU converts the crankshaft signal into a sequence of pulses, with each pulse representing 0.1° of crankshaft rotation:

1. When the TPU detects an edge of the crankshaft signal, TCR1 is read. TCR1 measures the time between two consecutive edges of the crankshaft signal, that is, the period.
2. Generation of equidistant 0.1° pulses for the current period starts. However, the pulse frequency is derived from the length of the previous period, that is, the current period is assumed to equal the preceding one.

   The number of 0.1° pulses depends on the angular distance between the detected edge and the edge that is to come. The number of pulses does not depend on the crankshaft speed. Higher crankshaft speed, for example, means smaller time intervals between the 0.1° pulses.

Accelerated crankshaft If the current period is actually smaller than the preceding one (acceleration of the crankshaft) not all 0.1° pulses can be generated within the assumed period. The ACU remembers the "lost" pulses and generates them in addition to the "normal" pulses in the next period(s).

Decelerated crankshaft If the current period is actually larger than the preceding one (deceleration of the crankshaft), the 0.1° pulses are generated within the assumed period. No more pulses are generated until the next period, as usual.

Pulse generation during a crankshaft gap Pulse generation during a crankshaft gap is basically the same as pulse generation during a crankshaft tooth. As the ACU knows the specification of the crankshaft wheel, it knows which two consecutive edges of the crankshaft signal precede a gap. The ACU takes their period as a reference and computes a theoretical "gap period". Then the ACU generates the pulses. When the first rising edge after the gap is detected, the actual "gap period" is measured and the subsequent period is computed. The pulses for the subsequent period are generated. Acceleration and deceleration is handled in the same way as for the ordinary pulse generation.

Time Counter Register TCR2

From the ACU, the 0.1° pulses are routed to the angle counter (time counter register TCR2), which is part of the MPC 565 processor of the RapidPro Control Unit.

TCR2 just counts the 0.1° pulses. The range of TCR2 is 0° … 719.9°. After 719.9°, TCR2 switches to 0° and starts again. The range of TCR2 reflects the angle range of the crankshaft (0° … 719.9°, at a resolution of 0.1°), that is, two crankshaft revolutions.

The initial angle value of TCR2 is specified by the Start angle parameter. Refer to RPCU_CRANK_SETUP_TPU_BLx. The initial angle value is zero by default.

For details on the TCR2, refer to Timer Basics on page 53.

Related topics

Basics

Basics on Crankshaft Sensor Signals	142
Basics on Speed Measurement (Crank Parameters)	215
Processing the Camshaft Signal	148
Synchronization Details on Parameter-Based Crankshaft Wheel Specifications	158
Timer Basics	53

Processing the Camshaft Signal

Objective

This section provides basic information on processing the camshaft signal so that it can be used by the RapidPro engine control features.

Signal conditioning via SC modules

The incoming camshaft signal must be connected to an SC module. The SC module must be mounted on the RapidPro Control Unit, not on a RapidPro SC Unit mounted on top of the RapidPro Control Unit. Both low and high active camshaft signals can be processed. It is not necessary to invert them externally, for example, via ConfigurationDesk for RapidPro.

Note

If the crankshaft wheel does not have a gap, note the following points:

- The SC module which the camshaft signal is connected to must be mounted on the RapidPro Control Unit, not on a RapidPro SC Unit mounted on top of the RapidPro Control Unit.
- An external inversion of the camshaft signal, for example, by ConfigurationDesk for RapidPro, is even not allowed. This would make synchronization impossible.

SC modules let you scale the camshaft signal according to your needs. Refer to Configuring RapidPro Hardware (ConfigurationDesk for RapidPro - Guide 📖). They can also perform noise filtering on an incoming camshaft signal (if implemented).

You have to use different SC modules depending on the type of the camshaft signal. For detailed information on the SC modules. Refer to the *RapidPro System Installation and Configuration* document.

Cam signal routing

From the SC module, the signal is routed to the time processor unit (TPU). If the crankshaft wheel does not have a gap, the camshaft signal is automatically routed to the ACU in addition to being routed to the TPU. This is necessary because the camshaft signal edge detected first is used to synchronize the angle measurement (TCR2).

Note

Only the parameter specification approach allows crank wheels without a gap, refer to Basics on Crankshaft Wheel Specification on page 201.

Filter for invalid signal edges

Erroneous edges in the camshaft signal, for example, due to noise, corrupt the measurement of the camshaft phase shift. You can filter out such invalid signal edges, refer to Basics on Camshaft Phase Shift Measurement on page 234.

Related topics

Basics

Synchronization Details on Wavetable-Based Crankshaft Wheel Specifications

Objective

To identify the exact position (stroke) of a four-stroke piston engine. Synchronizing the crankshaft and camshaft signals means identifying the 0° position of the angle counter.

Note

This chapter refers to a crankshaft wheel specification that is based on a wavetable, not on parameters. For details, refer to Basics on Crankshaft Wheel Specification on page 201.

Where to go from here

Information in this section

Information in other sections

Provides details on a crankshaft wheel specification based on wheel parameters instead of a wheel wavetable.

Basics on the Synchronization of the Crankshaft and Camshaft Signals (Wavetable-Based)

Objective

Explains how the 0° position of the crankshaft angle measurement is identified and the role of the camshaft signal.

Definition of the 0° position

By default, the 0° position of the angle measurement is internally defined to match the first wavetable data point. You can change this relationship by specifying a **Start angle** > 0°. All angle values relate to the 0° position.

Note

For the wavetables that come with the dSPACE software, the first data point relates to the first rising edge after a gap.

Engine offset If the 0° position of your engine does not match the first wavetable data point, you can specify an offset. The offset must be applied to specify any absolute angle values, for example, camshaft angles, TDCs, and interrupts. For details on implementing an engine offset, refer to How to Set Up Engine Offsets on page 209.

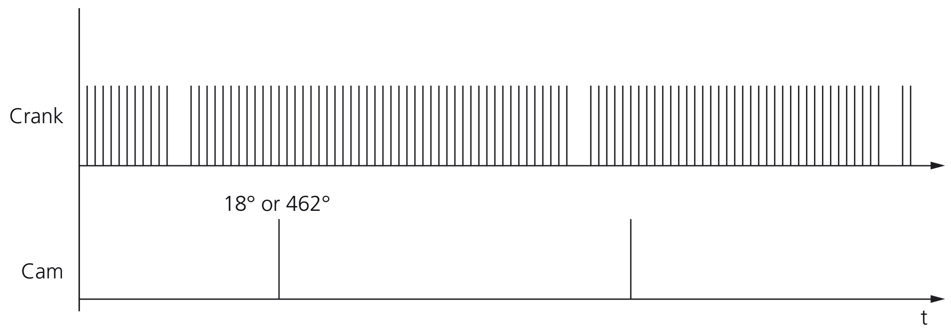
Crankshaft/camshaft wheel specification

Suppose the crankshaft and camshaft wheels have the following characteristics:

Wheel	Specification
Crankshaft	▪ Number of crankshaft teeth: 60 ▪ Missing teeth per gap: 2 ▪ Number of gaps: 1 ▪ Edge polarity: Rising edges ▪ Start angle: 0° (e.g., the 0° position equals the first rising edge after the gap) For details on the parameters, refer to Parameters Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Camshaft	▪ Number of camshaft pulses: 2 ▪ Vector of camshaft start angles (rising edges): [18°, 462°] ▪ Vector of camshaft end angles (falling edges): [54°, 489°] ▪ Edge polarity: Rising edges For details on the parameters, refer to Parameters Page (RPCU_CAM_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Synchronization of crankshaft and camshaft signals

Suppose the following crankshaft and camshaft signals (rising edges) were measured:

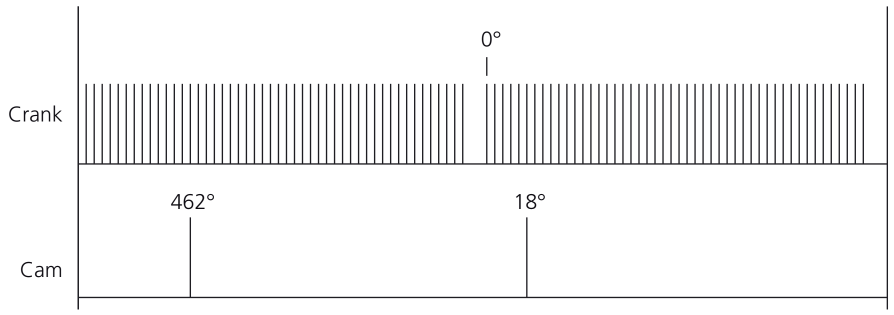


The question now is: Where is the 0° position so that the measured crankshaft and camshaft signals match the crank wheel and cam wheel specifications? Or in other words: Does the camshaft signal edge that was measured first correspond to 18° or 462° ?

There is only one solution for assigning the 0° position to a crankshaft signal edge so that the crankshaft and camshaft signals match the crankshaft/camshaft wheel specifications:

Note

The camshaft wheel markers must be unambiguously defined so that only one match exists.

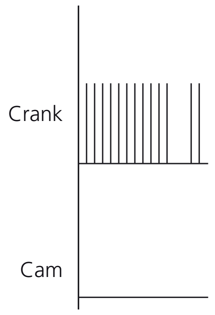


Synchronization process

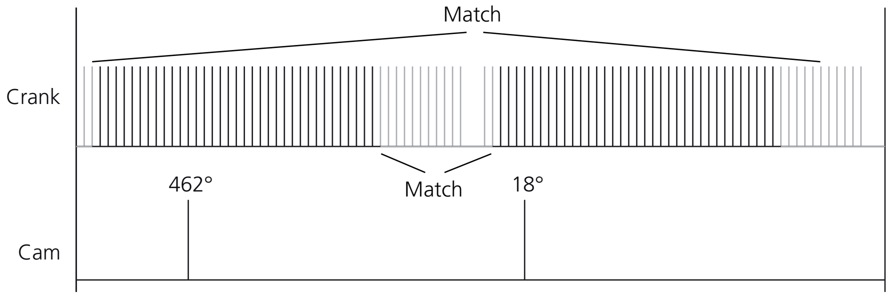
When the crankshaft and camshaft signals are measured, the system repeatedly tries (whenever a signal edge is detected) to match the signal edge pattern measured so far with the specified signal edge pattern. The more signal data has been collected, the more likely is an unambiguous match, and therefore synchronization.

Example Two signal match trials are described below.

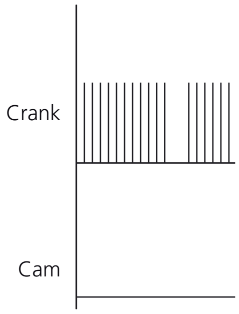
1. Suppose the following signal data has been collected so far (first trial):



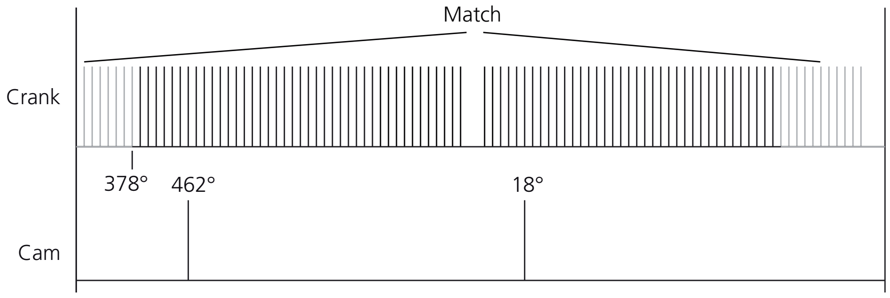
This leads to two matches with the specification, hence, synchronization is not achieved.



2. Suppose some more signal data has been collected (second trial):



Synchronization is achieved when the ACU detects the nonexistent camshaft signal edge which is "located" at 18° after the gap. The signal measured so far belongs to the stroke that relates to the 462° camshaft marker. Hence, the current crankshaft angle is computed to 378°, and the MPC565 microprocessor initializes the TCR2 angle counter to 378°.

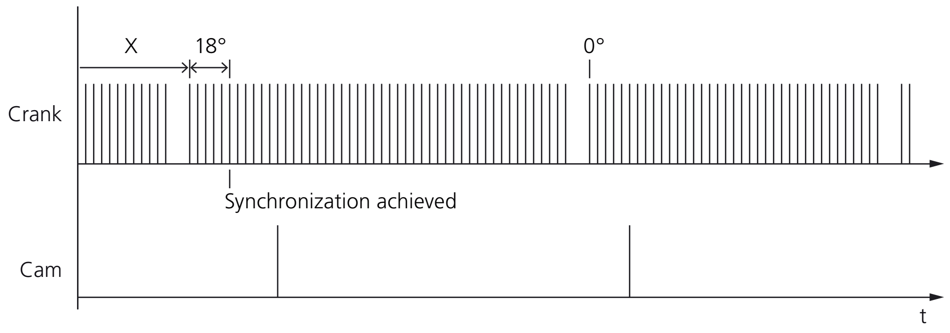


Tip

It is worth considering existing and nonexistent camshaft signal edge , refer to "Detecting nonexistent camshaft signal edges" below.

Synchronization time For engine control, it is important to achieve synchronization as quickly as possible. For example, the generation of injection and ignition pulses cannot start until the crankshaft and camshaft signals are synchronized.

The actual synchronization time depends on the crankshaft and camshaft wheels used (or their specifications) and the positions of the crankshaft and the camshaft at start time (Time = 0). For the above example of measured crankshaft/camshaft signals, synchronization is achieved at 18°+X after start time.



Worst-case synchronization time X depends on the engine position at start time plus (due to technical restrictions) 2 or 3 teeth. A worst-case synchronization time can be calculated for each crankshaft/camshaft wheel design according to the most unfavorable engine position at start time. Refer to Examples of Worst-Case Synchronization Times on page 155.

Impact of the crankshaft wheel specification

Collecting camshaft signal data The measurement of the camshaft signal is restricted to a specific crankshaft angle range depending on how the crankshaft wheel is specified:

- If the crankshaft wheel is specified by parameters, the camshaft signal is measured only within the user-defined camshaft evaluation segment, refer to Parameters Page (RPCU_CAM_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

  Note

  An inappropriate camshaft evaluation segment can cause synchronization problems. For an example, see Example #1 of a Too Small Camshaft Evaluation Segment on page 169.

- If the crankshaft wheel is specified by a wavetable, the specification of a camshaft evaluation segment is not necessary.

Detecting nonexistent camshaft signal edges Nonexistent camshaft signal edges are also considered:

- If the crankshaft wheel is specified by a wheel wavetable, nonexisting camshaft signal edges are always detected because Force synchronization without camshaft signal edge detection is always enabled and you cannot disable it.

  Note

  Force synchronization without camshaft signal edge detection could lead to a temporary (720° at maximum) synchronization failure if any wiring is defective or the camshaft marker specification does not meet the actual camshaft wheel design.

- If the crankshaft wheel is specified by parameters, nonexistent camshaft signal edges are detected if you select the Force synchronization without camshaft signal edge detection option, refer to Advanced Page (RPCU_CAM_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

NOTICE

Risk of damage to the combustion engine

Using Force synchronization without camshaft signal edge detection could lead to a synchronization failure if any wiring is defective or an improper synchronization segment was selected.
Check the wiring harness and the **Segment specification** (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Related topics

HowTos

Examples

References

RPCU_ENG_STATUS_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_ENG_STATUS_WT_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Examples of Worst-Case Synchronization Times

Objective

Explains the worst-case synchronization time for certain crankshaft and camshaft wheel designs.

Worst-case synchronization time

Worst-case synchronization time means the largest possible angle range after which synchronization is achieved. The worst case relates to the following circumstances:

- Inappropriate crankshaft/camshaft wheel design
- Unfavorable engine position at start time

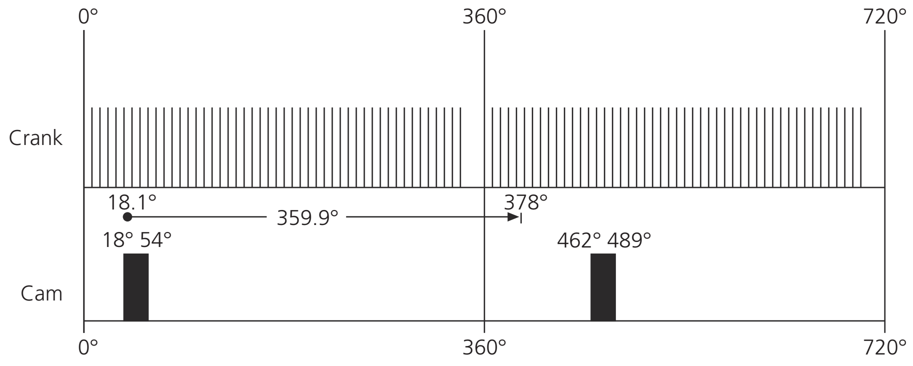
The following examples present the worst-case synchronization time for a selection of crankshaft/camshaft wheel designs. The wheel design is represented by a plot of the synchronized crank and cam signals.

Note

The following examples are idealized situations as they assume a camshaft tolerance of 0.

Example 1

Cam signal: Rising edge. Synchronization is achieved after 359.9° at the latest (359.9° = (360°-18.1°) + 18°). The most unfavorable engine start position is at 18.1°.

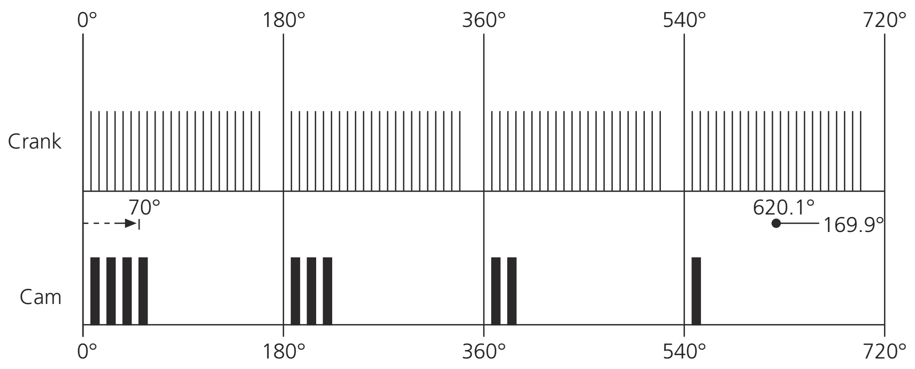


Example 2

Cam signal: Either edges. Synchronization is achieved after 169.9° = (720° - 620.1°) + 70° at the latest. The most unfavorable engine start position is at 620.1°.

Cam marker specification (from left to right in the illustration below):

- #1: 10° … 20°
- #2: 30° … 40°
- #3: 50° … 60°
- #4: 70° … 80°
- #5: 190° … 200°
- #6: 210° … 220°
- #7: 230° … 240°
- #8: 370° … 380°
- #9: 390° … 400°
- #10: 550° … 580°


0°
180°
360°
540°
720°
Crank
70°
620.1°
169.9°
Cam
0°
180°
360°
540°
720°

Synchronization Details on Parameter-Based Crankshaft Wheel Specifications

Objective

Note

This chapter refers to a crankshaft wheel specification that is based on parameters instead of a wavetable. For details, refer to Basics on Crankshaft Wheel Specification on page 201.

Synchronization of the angle counter (time-counter register TCR2) is a crucial internal process. The camshaft signal is evaluated to assign the 0° position to TCR2, the range is 0° … 719.9°. TCR2 is triggered by the crankshaft signal. You should be familiar with this process when you set up the measurement of the crankshaft and camshaft signals.

Definition of the 0° position

By default, the 0° position of the angle measurement is internally defined to match the first relevant tooth edge (rising or falling) after a crankshaft wheel gap. You can change this relationship by specifying a **Start angle** > 0°. All angle values relate to the 0° position.

Engine offset If the 0° position of your engine does not match the first edge of the crankshaft signal after a gap, you can specify an offset. The offset must be applied to specify any absolute angle values, for example, camshaft angles, TDCs, and interrupts.

Note

Edges of the camshaft signal that come slightly before the first crankshaft tooth after the gap cannot be detected until nearly a complete camshaft revolution is performed, even if the engine's 0° position comes before them.

Where to go from here

Information in this section

Information in other sections

Synchronization With Crankshaft Gap(s)

Objective

This section provides basic information on the synchronization of the angle counter (TCR2) with the camshaft signal if the crankshaft wheel has one or more gaps.

Synchronization process

Synchronization of the angle measurement is an internal process and starts automatically after the engine has started. It comprises the following steps:

1. As soon as a crankshaft gap is detected, the angle counter is set to the user-defined start value (default: 0°) and angle measurement starts with a preliminary 0° position.

   As soon as a crankshaft gap is detected, the following feature is available:

   - Measurement of the engine's speed

   Refer to Triggering the Angle Counter via Gap Detection on page 161.
2. Within a user-defined angle range (camshaft evaluation segment), the angle values relating to the edges of the camshaft signal are logged.

   Refer to Logging the Camshaft Signal Within the Evaluation Segment on page 162.
3. The logged camshaft angle values are compared to the angle values of the camshaft marker specification, and angle measurement is synchronized with the camshaft marker specification.

   As soon as angle measurement is synchronized, the following features are available:

   - Measurement of the engine's position
   - Generation of injection and ignition pulses

Note

The end angle of the camshaft evaluation segment determines when synchronization ends, if it is successful. Refer to Segment end angle (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Failure of synchronization

Synchronization may fail, for example, due to an ambiguous camshaft marker specification (refer to Example of an Ambiguous Camshaft Marker Specification on page 177), or due to a too small camshaft evaluation segment (refer to Example #1 of a Too Small Camshaft Evaluation Segment on page 169).

If multiple camshaft wheels are used, at least one signal edge must have been detected within the camshaft evaluation segment for each camshaft wheel. This prevents erroneous synchronization caused, for example, by an interrupted signal line.

If synchronization fails, you have to check your specifications. Refer to RPCU_CRANK_SETUP_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮) and RPCU_CAM_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Note

To check whether synchronization was successful, you must monitor the engine status information. Refer to RPCU_ENG_SPEED_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Where to go from here

Information in this section

Information in other sections

Triggering the Angle Counter via Gap Detection

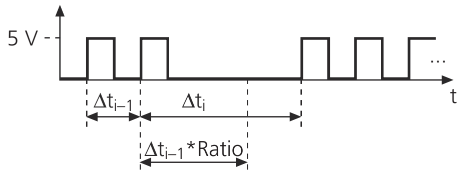
Objective

Triggering the angle counter (TCR2) is part of the synchronization process. This section provides basic information on making use of gap detection as the trigger for angle measurement.

Gap detection algorithm

Angle measurement is automatically triggered as soon as a crankshaft gap is detected after the engine was started.

A crankshaft gap is found as soon as the ACU detects a period Δt_i greater than (Δt_{i-1} * Ratio). For details on the Ratio parameter. Refer to Basics on Speed Measurement (Crank Parameters) on page 215.



Note

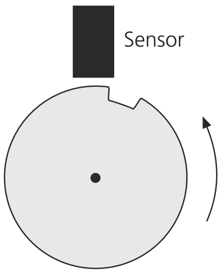
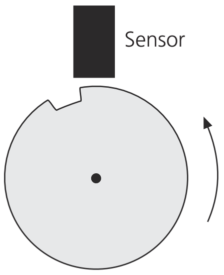
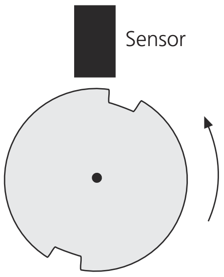
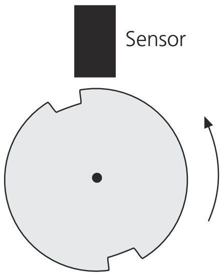
In the synchronized state, the ACU knows the position of a crankshaft gap, as this can be derived from the crankshaft and camshaft specifications. If a tooth is detected instead of a gap, synchronization is lost (refer to Error Handling on page 268).

Advantage of multiple gaps

The more gaps a crankshaft wheel has, the less time is needed, on average, to detect a gap for the first time. See the example below.

Example

Suppose you have two crankshaft wheels. One wheel has one gap, the other has two gaps. The following table shows how long it can take until a gap is detected:

Number of Gaps	Time for Gap Detection:	
	Best Case	**Worst Case**
One	 immediately	 after about one revolution
Two	 immediately	 after about half a revolution

Thus, multiple gaps optimize the worst-case detection time, but best-case is not affected.

Related topics

Basics

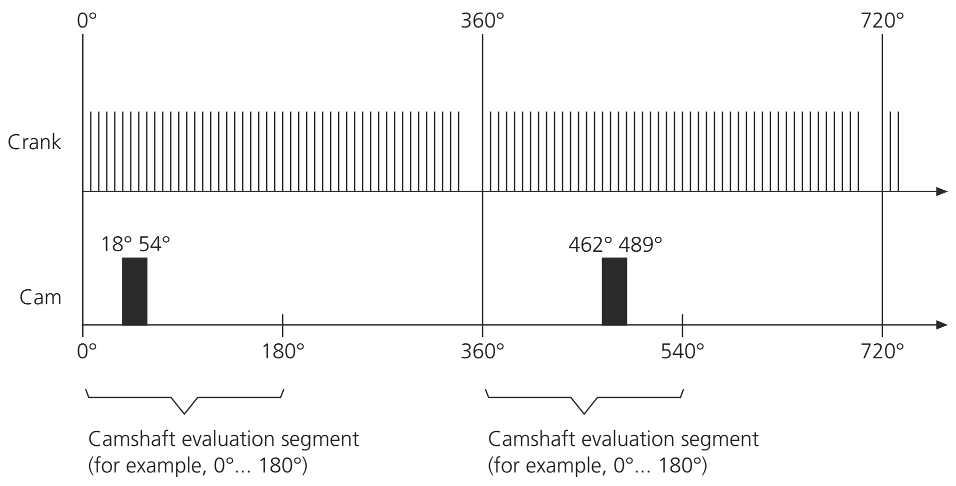
Logging the Camshaft Signal Within the Evaluation Segment

Objective

Logging the camshaft signal is part of the synchronization process. This section provides basic information on the logging algorithm.

Camshaft evaluation segment

Camshaft angle values can be measured only within the user-defined camshaft evaluation segment. The camshaft evaluation segment is specified by a segment start angle and a segment end angle.


0°
360°
720°
Crank
18° 54°
462° 489°
Cam
0°
180°
360°
540°
720°
Camshaft evaluation segment (for example, 0°... 180°)
Camshaft evaluation segment (for example, 0°... 180°)

When an edge of the camshaft signal is detected, the angle counter is read, and the current TCR2 angle value is logged. (In the illustration above, edge polarity is set to “Both edges”.)

For details on an appropriate specification of these angles, refer to the parameters that relate to Segment specification (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

Edge polarity

The detection of both rising and falling edges is the default setting (“Either edges”). It is also possible to detect only rising or only falling edges. Refer to Edge polarity (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

No gap detection

If a crankshaft gap is detected again within the camshaft evaluation segment, angle measurement is not restarted.

Related topics

Basics

Triggering the Angle Counter via Gap Detection.. 161

Examples

Example #1 of a Too Small Camshaft Evaluation Segment.. 169
Example of a Too Large Camshaft Evaluation Segment.. 166
Example of Synchronization With Camshaft Phase Shift... 196

Updating the Angle Counter with the Camshaft Marker Specification

Objective

Updating the angle counter is part of the synchronization process. This section provides basic information on the update algorithm.

Reasons for an update

Angle counter and camshaft marker specification must relate to the same 0° position. If the preliminary 0° position is invalid, the angle counter has to be corrected.

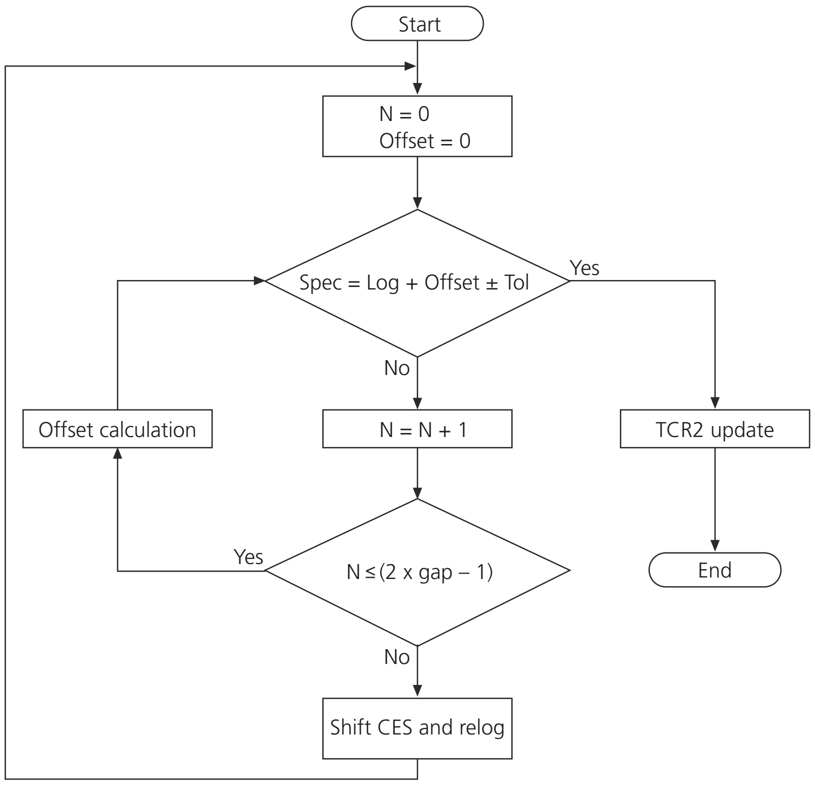
Definition of top dead center

All angle values of the injection and ignition pulses relate to the cylinder-specific top dead center (TDC). A TDC is the reversal point of a piston after compression and exhaust, and is represented by an angle value relative to the 0° position. TDCs are design parameters of an internal combustion engine which you must know.

Update of the angle counter

The angle counter (TCR2) is updated as follows:

Log	Vector of the logged angle values
Spec	Vector of the camshaft marker angle values
Offset	Offset applied to TCR2 (internal algorithm)
Tol	User-defined camshaft tolerance
Gap	Number of crankshaft wheel gaps
CES	User-defined camshaft evaluation segment



1. The logged angle values (Log) are compared to the specification of the camshaft markers (Spec):

   ```
   Spec = Log ± Tolerance
   ```

   The user-defined Camshaft tolerance (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖) is used for this comparison.

   If the comparison finds a match, synchronization is finished.
2. If the comparison does not find a match, an offset is calculated, taking the segmentation of the crankshaft wheel (refer to Symmetric crankshaft wheels on page 144) into consideration, as shown by the following pseudo-code:

   ```
   Offset(N) = N * (360° / Number of gaps)
      N = 1 … (2 * Number of gaps - 1)
   ```

   Subsequently, the comparison is performed once more:

   ```
   Spec = Log + Offset ± Tolerance
   ```
3. If all offset possibilities have been exhausted but no match was found (`N > 2*Gap - 1`), the logged angle values are cleared, the camshaft evaluation segment (CES) is shifted taking the segmentation of the crankshaft wheel (refer to Symmetric crankshaft wheels on page 144) into consideration, and the camshaft signal is logged once more:

   ```
   CamshaftEvaluationSegment Shift (StartAngle, EndAngle)
      StartAngle = StartAngle + (360° / Gaps)
      EndAngle = EndAngle + (360° / Gaps)
   End
   ```

Note

No angle values are returned and no injection and ignition pulses are generated until synchronization is successful.
If the comparison does not find a match at all, the synchronization process is an infinite loop.

Related topics

Basics

Example of a Too Large Camshaft Evaluation Segment

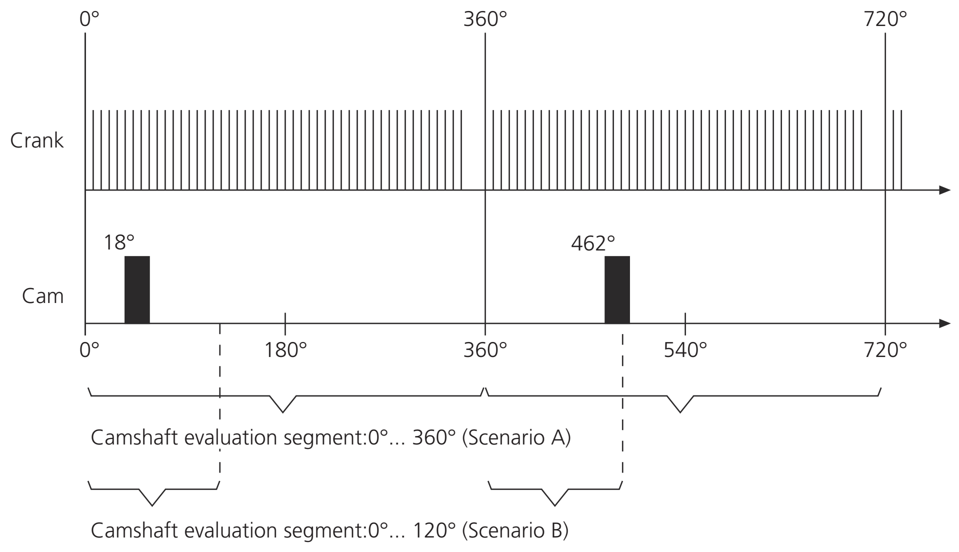
Introduction

This example shows that the length of the camshaft evaluation segment is the deciding factor for the time required until synchronization is achieved.

Suppose the crankshaft and camshaft parameters are as follows:

Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth
Camshaft	Markers: ▪ #1: 18° … 54° (1st crankshaft revolution) ▪ #2: 462° … 489° (2nd crankshaft revolution) Edge polarity: “Rising edge”

The following crankshaft and camshaft signals are generated:


0°
360°
720°
Crank
18°
462°
Cam
0°
180°
360°
540°
720°
Camshaft evaluation segment:0°... 360° (Scenario A)
Camshaft evaluation segment:0°... 120° (Scenario B)

Scenarios

There are two scenarios in this example:

Scenario	Camshaft Evaluation Segment
A	0° … 360° (too large)
B	0° … 120° (ok)

Scenario A, marker #1

Suppose the engine starts in such a position that camshaft marker #1 is detected first (18°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle value now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	18°	Yes	Yes

The logged camshaft angle value matches the specified camshaft angle value. Synchronization is achieved after 360°, as this is the size of the camshaft evaluation segment.

Scenario A, marker #2

Suppose the engine starts in such a position that camshaft marker #2 is detected first (462°). The camshaft angle values are logged as follows (102° = 462°-360°):

Logged Edges	
Rising	**Falling**
102°	–

Comparing the logged camshaft angle value with the specified camshaft angle value now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	102°	No	No

The logged camshaft angle value does not match the specified camshaft angle value. Synchronization is not achieved. Subsequently, the internal algorithm corrects the logged angle value (+360°). Comparing the logged camshaft angle value (462°) with the specified camshaft angle value now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	462° (corrected)	Yes	Yes

The logged camshaft angle value matches the specified camshaft angle value. Synchronization is achieved after 360°, as this is the size of the camshaft evaluation segment. The angle counter (TCR2) is updated (+360°).

Scenario B, marker #1

Suppose the engine starts in such a position that camshaft marker #1 is detected first (18°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle value now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°	18°	Yes	Yes

The logged camshaft angle value matches the specified camshaft angle value. Synchronization is achieved after 120°, as this is the size of the camshaft evaluation segment.

Scenario B, marker #2

Suppose the engine starts in such a position that camshaft marker #2 is detected first (462°). The camshaft angle values are logged as follows (102° = 462°-360°):

Logged Edges	
Rising	**Falling**
102°	–

Comparing the logged camshaft angle value with the specified camshaft angle value now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	102°	No	No

The logged camshaft angle value does not match the specified camshaft angle value. Synchronization is not achieved. Subsequently, the internal algorithm corrects the logged angle value (+360°). Comparing the logged camshaft angle value (462°) with the specified camshaft angle value now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	462° (corrected)	Yes	Yes

The logged camshaft angle value matches the specified camshaft angle value. Synchronization is achieved after 120°, as this is the size of the camshaft evaluation segment. The angle counter (TCR2) is updated (+360°).

Conclusion from this example

A camshaft evaluation segment that is too large wastes time (delay of injection and ignition pulse generation). You should therefore adapt the camshaft evaluation segment to the specified camshaft angle values.

Related topics

Examples

Example #1 of a Too Small Camshaft Evaluation Segment

Introduction

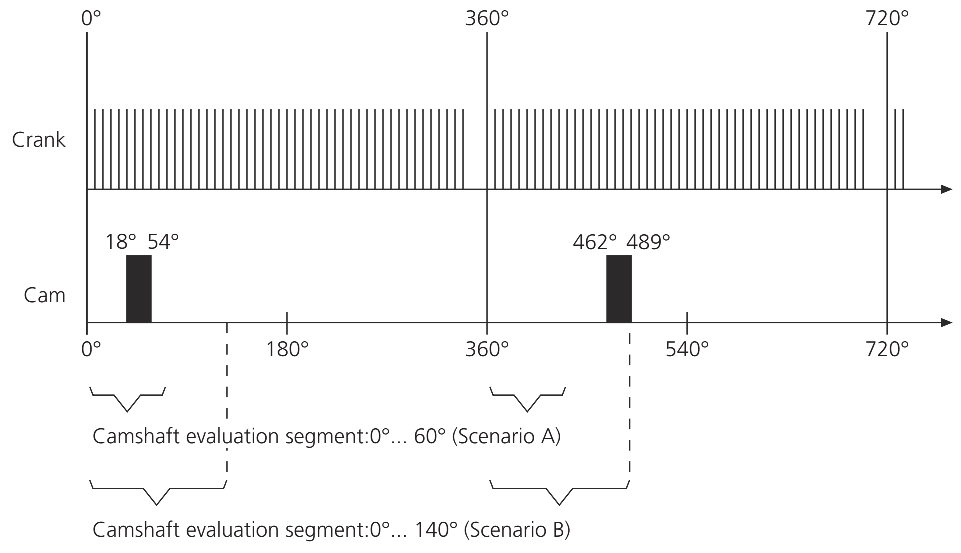
This example shows that a too small camshaft evaluation segment prolongs the synchronization phase.

Suppose the crankshaft and camshaft parameters are as follows:

Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth
Camshaft	Markers: ▪ #1: 18° … 54° (1st crankshaft revolution)

Wheel	Parameters
	▪ #2: 462° … 489° (2nd crankshaft revolution) Edge polarity: "Either edges" (rising and falling edge)

The following crankshaft and camshaft signals are generated:


0°
360°
720°
Crank
Cam
18° 54°
462° 489°
0°
180°
360°
540°
720°
Camshaft evaluation segment:0°... 60° (Scenario A)
Camshaft evaluation segment:0°... 140° (Scenario B)

Scenarios

There are two scenarios in this example:

Scenario	Camshaft Evaluation Segment
A	0° … 60° (too small)
B	0° … 140° (ok)

Scenario A, marker #1

Suppose the engine is in such a position that camshaft marker #1 is detected first (18°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	54°

Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°/54° 462°/489°	18°/54°	Yes	Yes

The logged camshaft angle values match the specified camshaft angle values. Synchronization is achieved after 60°, as this is the size of the camshaft evaluation segment.

Scenario A, marker #2

Suppose the engine is in such a position that camshaft marker #2 is detected first (462°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
–	–

No angle values are logged, because at 102°/129° (462°/489° minus 360°) they are outside the camshaft evaluation segment (0° … 60°).

The next time the crankshaft gap is detected (next crankshaft revolution), the camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
378°	414°

Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°/54° 462°/489°	378°/414°	No	No

The logged camshaft angle values do not match the specified camshaft angle values. Synchronization is not achieved. Subsequently, the internal algorithm corrects the logged angle value (+360°). A new comparison of the logged camshaft angle values with the specified camshaft angle values returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°/54° 462°/489°	18°/54° (corrected)	Yes	Yes

The logged camshaft angle values match the specified camshaft angle values. Synchronization is achieved after 420° (360° + 60°), which is one crankshaft revolution plus the camshaft evaluation segment. The angle counter (TCR2) is updated (+360°).

Scenario B, marker #1

Suppose the engine is in such a position that camshaft marker #1 is detected first. The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	54°

Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°/54° 462°/489°	18°/54°	Yes	Yes

The logged camshaft angle values match the specified camshaft angle values. Synchronization is achieved after 140°, as this is the size of the camshaft evaluation segment.

Scenario B, marker #2

Suppose the engine is in such a position that camshaft marker #2 (462°) is detected first. The camshaft angle values are logged as follows:

Logged Edges	
Rising	Falling
102°	129°

Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°/54° 462°/489°	102°/129°	No	No

The logged camshaft angle values do not match the specified camshaft angle values. Synchronization is not achieved. Subsequently, the internal algorithm corrects the logged angle value (+360°). A new comparison of the logged camshaft angle values with the specified camshaft angle values returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°/54° 462°/489°	462°/489° (corrected)	Yes	Yes

The logged camshaft angle values match the specified camshaft angle values. Synchronization is achieved after 140°, as this is the size of the camshaft evaluation segment. The angle counter (TCR2) is updated (+360°).

Conclusion from this example

If the camshaft evaluation segment does not cover the angle values of all the specified camshaft markers, this can result in one or even more additional crankshaft revolutions until synchronization is achieved.

Related topics

Examples

Example #2 of a Too Small Camshaft Evaluation Segment

Introduction

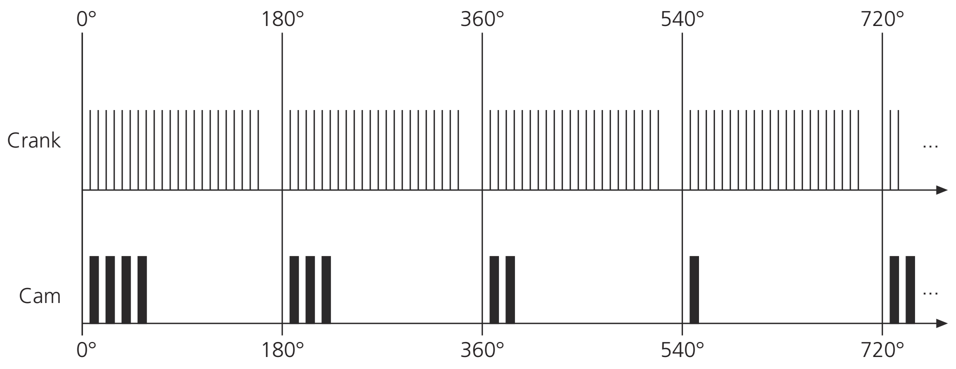
This example shows that a too small camshaft evaluation segment can lead to an erroneous synchronization, if:

- The size of the camshaft evaluation segment equals the size of the crankshaft segment, and
- The camshaft markers projected on the same camshaft evaluation segment are congruent.

Suppose the crankshaft and camshaft parameters are as follows:

Wheel	Parameters
Crankshaft	60 – 2 * 2 wheel: ▪ 60 teeth ▪ Two gaps, each with two missing teeth
Camshaft	Markers: ▪ #1: 10° … 20° ▪ #2: 30° … 40° ▪ #3: 50° … 60° ▪ #4: 70° … 80° ▪ #5: 190° … 200° ▪ #6: 210° … 220° ▪ #7: 230° … 240° ▪ #8: 370° … 380° ▪ #9: 390° … 400° ▪ #10: 550° … 580° Edge polarity: "Either edges" (rising and falling edges)

The following crankshaft and camshaft signals are generated:



Scenarios

There are two scenarios in this example:

Scenario	Camshaft Evaluation Segment
A	0° … 180° (too small)
B	0° … 360° (ok)

Scenario A, marker #10

Suppose the camshaft evaluation segment is set to 180°, and the engine is in such a position that camshaft marker #10 is detected first (550°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
10°	20°

Comparing the logged camshaft angle values with the specified camshaft angle values (from #1 to #10) now returns the following:

Specified Angles	Logged Angles	Match	Synch.
10°/20° (#1) 30°/40° (#2) 50°/60° (#3) 70°/80° (#4) 190°/200° (#5) 210°/220° (#6) 230°/240° (#7) 370°/380° (#8) 390°/400° (#9) 550°/560° (#10)	10°/20°	Yes	Yes

The logged camshaft angle values match the specified camshaft angle values (10°/20°). Synchronization is achieved after 180°. However, this synchronization is erroneous, as marker #10 is incorrectly identified as marker #1.

Scenario B, marker #10

Suppose the camshaft evaluation segment is set to 360°, and the engine is in such a position that camshaft marker #10 is detected first (550°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
10°	20°
190°	200°
210°	220°
230°	240°
250°	260°

Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
10°/20° (#1)	10°/20°	Yes	No
30°/40° (#2)	190°/200°	No	
50°/60° (#3)	210°/220°	No	
70°/80° (#4)	230°/240°	No	
190°/200° (#5)	250°/260°	No	
210°/220° (#6)			
230°/240° (#7)			
370°/380° (#8)			
390°/400° (#9)			
550°/560° (#10)			

Synchronization is not achieved, as the logged angle value pairs 190°/200° etc. do not match the camshaft marker specification.

Subsequently, an internal algorithm corrects the logged angle values (+180°). Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
10°/20° (#1)	190°/200°	Yes	No
30°/40° (#2)	370°/380°	No	
50°/60° (#3)	390°/400°	No	
70°/80° (#4)	410°/420°	No	
190°/200° (#5)	430°/440°	No	
210°/220° (#6)	(corrected)		
230°/240° (#7)			
370°/380° (#8)			
390°/400° (#9)			
550°/560° (#10)			

Synchronization is not achieved, as the logged angle value pairs 370°/380° etc. do not match the camshaft marker specification.

Subsequently, the internal algorithm corrects the logged angle values again (+180°). Comparing the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
10°/20°	370°/380°	Yes	No
30°/40°	550°/560°	No	
50°/60°	570°/580°	No	
70°/80°	590°/600°	No	
190°/200°	610°/620°	No	
210°/220°	(corrected)		
230°/240°			
370°/380°			
390°/400°			
550°/560°			
10°/20°			
30°/40°			

Synchronization is not achieved, as the logged angle value pairs 550°/560° etc. do not match the camshaft marker specification.

Subsequently, the internal algorithm corrects the logged angle values again (+180°). Comparing of the logged camshaft angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
10°/20°	550°/560°	Yes	Yes
30°/40°	10°/20°	Yes	
50°/60°	30°/40°	Yes	
70°/80°	50°/60°	Yes	
190°/200°	70°/80°	Yes	
210°/220°	(corrected)		
230°/240°			
370°/380°			
390°/400°			
550°/560°			
10°/20°			
30°/40°			
50°/60°			
70°/80°			

Synchronization is achieved (after 360°), as all the logged angle value pairs match the camshaft marker specification. The angle counter (TCR2) is updated (+540°).

Conclusion from this example

The camshaft evaluation segment must be large enough for an unambiguous sequence of angle values of the camshaft marker specification to come into focus. Only the angle values actually logged within the camshaft evaluation segment are searched for angle value matches with the camshaft marker specification.

Related topics

Examples

Example of an Ambiguous Camshaft Marker Specification

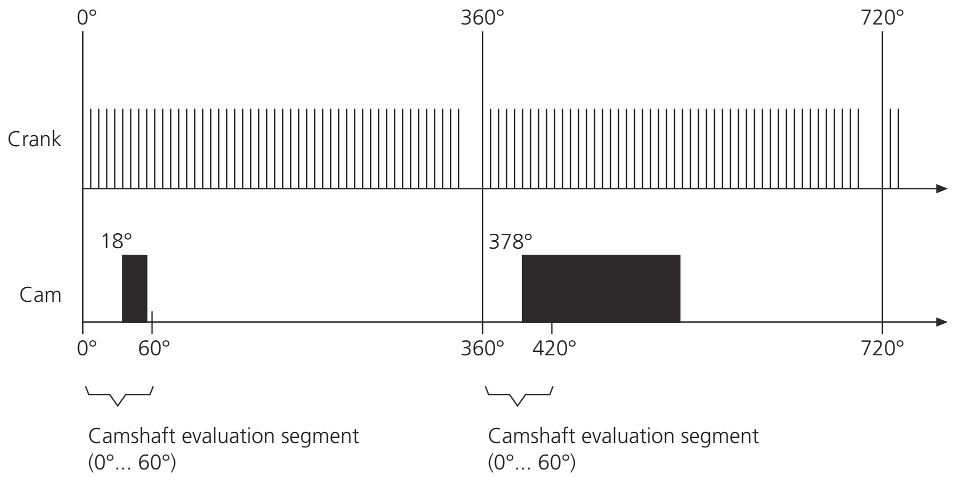
Use scenario

This example shows that ambiguous specification of camshaft markers can lead to invalid synchronization.

Suppose the crankshaft and camshaft parameters are as follows:

Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth
Camshaft	Markers: ▪ #1: 18° … 54° (1st crankshaft revolution) ▪ #2: 378° … 489° (2nd crankshaft revolution) Edge polarity: "Rising edges" Camshaft evaluation segment: 0° … 60°

The following crankshaft and camshaft signals are generated:


0°
360°
720°
Crank
18°
378°
Cam
0°
60°
360°
420°
720°
Camshaft evaluation segment (0°... 60°)
Camshaft evaluation segment (0°... 60°)

Marker #1

Suppose the engine is in such a position that camshaft marker #1 is detected first (18°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 378°	18°	Yes	Yes

Synchronization is achieved after 60°, as this is the size of the camshaft evaluation segment.

Marker #2

Suppose the engine is in such a position that camshaft marker #2 is detected first (378°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 378°	18°	Yes	Yes

Synchronization is achieved after 60° (as this is the size of the camshaft evaluation segment), but during the wrong crankshaft revolution, which means erroneous synchronization.

Conclusion from this example

In this example, it would be necessary to specify the edge polarity of the camshaft signal to “Falling edge”. “Either edges” would also be appropriate.

Note

The distance between the specified angle values (same edge type) of different camshaft markers must not match a multiple of the crankshaft segment.

Related topics

Examples

Example of the Benefit of a Camshaft Wheel with Multiple Camshaft Markers

Introduction

This example shows the influence of the number of camshaft markers on the minimum and maximum time required until synchronization is achieved.

Suppose the crankshaft and camshaft parameters are as follows:

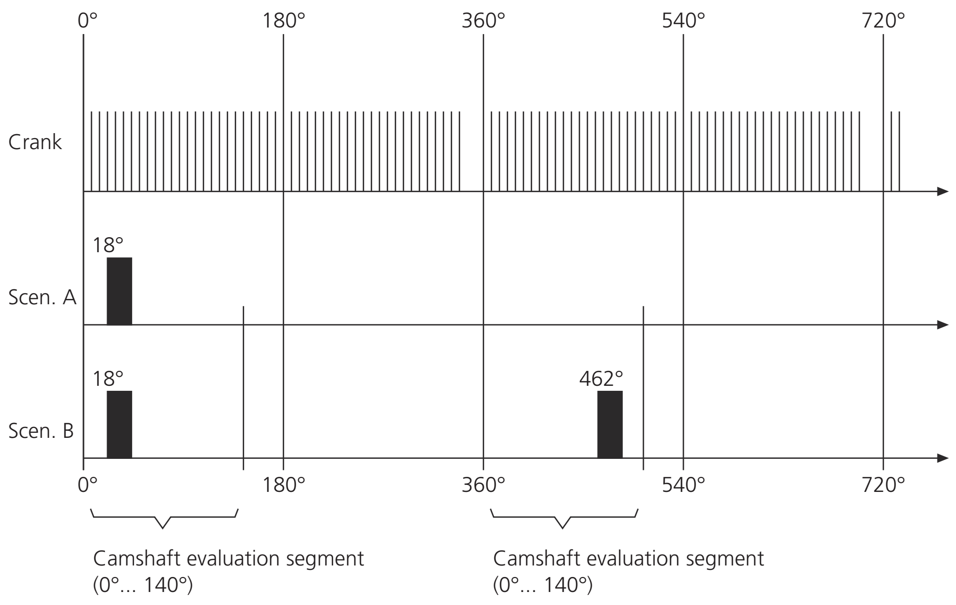
Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth (segment = 360°)
Camshaft	Edge polarity: "Rising edges" Camshaft evaluation segment: 0° … 140°

Scenarios

There are two scenarios in this example:

Scenario	Camshaft Markers
A	#1: 18° … 54° (1st crankshaft revolution)
B	#1: 18° … 54° (1st crankshaft revolution) #2: 462° … 489° (2nd crankshaft revolution)

The following crankshaft and camshaft signals are generated:



Scenario A, first gap

Suppose the engine is in such a position that the first crankshaft gap (actual 0° position) is detected first (0°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle values returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°	18°	Yes	Yes

Synchronization is achieved after 140°, as this is the size of the camshaft evaluation segment.

Scenario A, second gap

Suppose the engine is in such a position that the second crankshaft gap is detected first (360°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
–	–

No angle values are logged, because there is no camshaft marker to be detected.

Subsequently, the first crankshaft gap (actual 0°-position) is detected, and the camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18°	18°	Yes	Yes

Synchronization is achieved after 500° (360° + 140°), which is the additional crankshaft revolution (segment) plus the camshaft evaluation segment.

Scenario B, first gap

Suppose the engine is in such a position that the first crankshaft gap (actual 0° position) is detected first (0°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
18°	–

Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	18°	Yes	Yes

Synchronization is achieved after 140°, as this is the size of the camshaft evaluation segment.

Scenario B, second gap

Suppose the engine is in such a position that the second crankshaft gap is detected first (360°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
102°	–

Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	102°	No	No

The logged angle values do not match the specified angle values. Synchronization is not achieved.

Subsequently, the internal algorithm corrects the logged angle values (+360°). Comparing the logged camshaft angle value (462°) with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
18° 462°	462°	Yes	Yes

Synchronization is achieved after 140°, as this is the size of the camshaft evaluation segment. The angle counter (TCR2) is updated (+360°).

Conclusion from this example

The more camshaft markers there are, the less maximum time is required for synchronization. Injection and ignition pulse generation, for example, can start earlier. However, the minimum time is not affected. Which time is actually required within this range depends on the initial position of the engine, or crankshaft (0° position).

Note

However, to make use of multiple camshaft markers, your crankshaft wheel must have at least the same number of gaps. In other words, each crankshaft gap must precede one camshaft marker.

Related topics

Examples

Example #1 of a Too Small Camshaft Evaluation Segment.. 169
Example of a Too Large Camshaft Evaluation Segment.. 166
Example of an Ambiguous Camshaft Marker Specification.. 177

Example of Synchronization Without Camshaft Signal

Objective

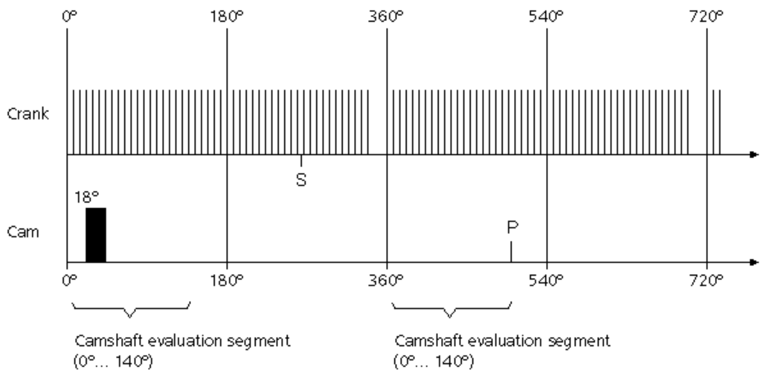
This example illustrates the Force synchronization without camshaft signal edge detection option including the possible risks.

Scenario

Suppose the crankshaft and camshaft parameters are as follows:

Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth (segment = 360°)

Wheel	Parameters
Camshaft	Marker: 18° … 54° (1st crankshaft revolution) Edge polarity: "Rising edges" Camshaft evaluation segment: 0° … 140°



Suppose the engine starts in position S. No camshaft angle values are logged within the camshaft evaluation segment as no signal edge occurs.

Because a camshaft marker actually exists fitting the angle range [0° ... 360°] the synchronization algorithm assumes that the camshaft evaluation segment here refers to the second crankshaft rotation [360° ... 720°]. Hence, the position P that is related to the end of the camshaft evaluation segment can be derived.

In this example, synchronization is possible at position P:

P = Segment end angle + 360°

Conclusion from this example

The angle counter can be quickly synchronized always during the first crankshaft rotation. Either a signal edge is actually detected, or the synchronization algorithm assumes that a signal edge will be detected during the next crankshaft rotation.

NOTICE

Risk of damage to the combustion engine

If the signal edge is not detected during the first crankshaft rotation though it had to be, synchronization is invalid (360°-shift error) and the combustion engine can be damaged.
Check the sensor and its wiring, and also the specification of the camshaft evaluation segment.

Possible reasons for a non-detection of camshaft signal edges are:
- Any kind of sensor defects, such as cable disruption or signal distortion.
- The camshaft evaluation segment does not cover the camshaft marker (wrong specification).

Related topics

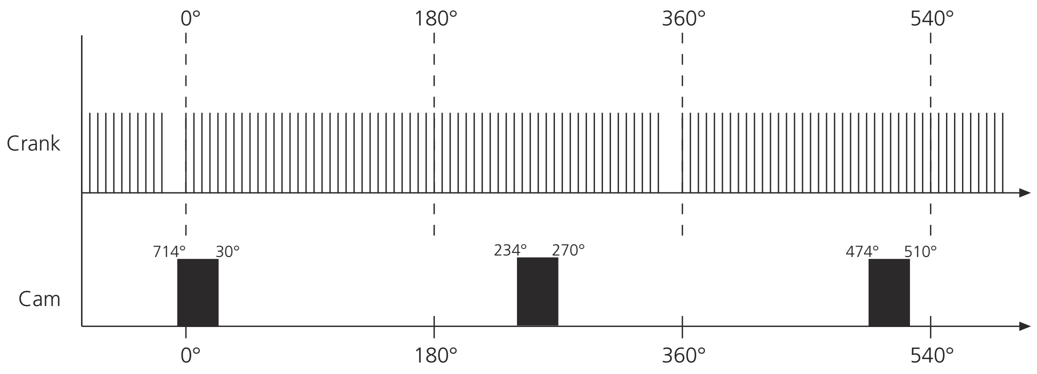
References

Advanced Page (RPCU_CAM_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference)

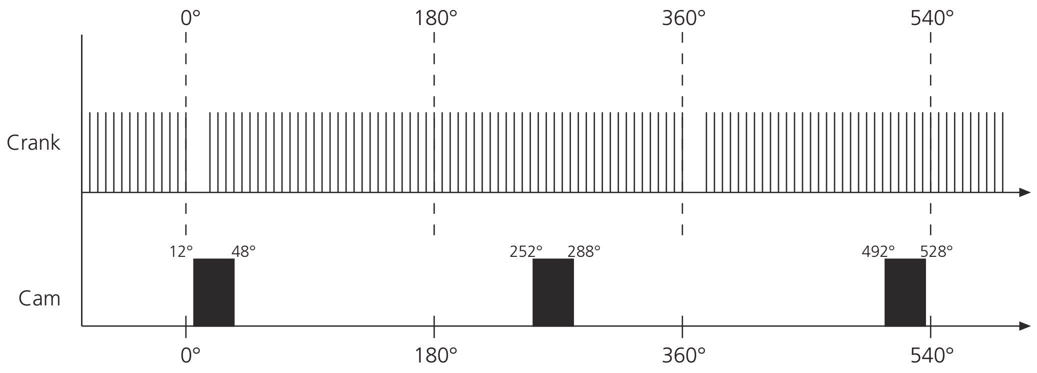
Example of Implementing an Engine Offset

Use scenario

Suppose, the following crankshaft and camshaft signals are generated:



Engine offset Suppose you prefer the last edge of the crankshaft signal before the gap to be the 0° position instead of the first edge after the gap. In consequence, the angle axes must be shifted 18° to the left (i.e., engine offset = +18°):

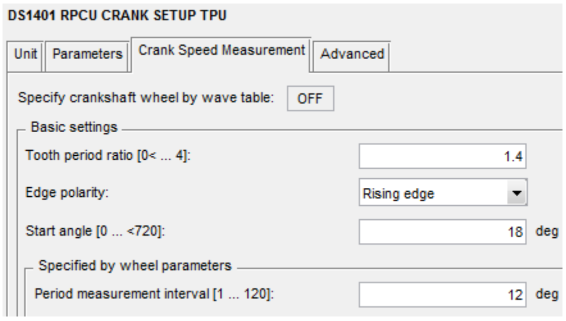


Note

In the above example, the angle shift (engine offset) causes the following:
- Change of the camshaft signal polarity at 0°, from "High" to "Low" (affects the **Signal polarity before first transition** parameter).
- The order of the rising edges of the camshaft markers changes: 12°, 252°, 492° instead of 234°, 474°, 714°, which affects the specification of the **Vector of camshaft start angles** parameter.

An implementation approach

In the dialog of the RPCU_CRANK_SETUP_TPU_BLx block, the following settings are chosen to implement the engine offset. The **Start angle** is determined by the engine offset (18°).


DS1401 RPCU CRANK SETUP TPU

Specify crankshaft wheel by wave table: OFF

Basic settings

Tooth period ratio [0< ... 4]: 1.4

Edge polarity: Rising edge

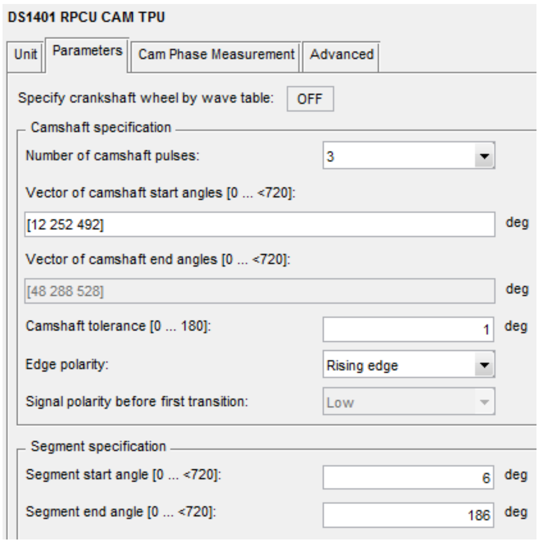
Start angle [0 ... <720]: 18 deg

Specified by wheel parameters

Period measurement interval [1 ... 120]: 12 deg

For the implementation instructions, refer to How to Set Up Engine Offsets on page 209.

In the dialog of the RPCU_CAM_TPU_BLx block, the following settings are chosen to implement the engine offset. The **Signal polarity before first transition** is not evaluated as the **Edge polarity** parameter is *not* set to "Either edge". The **Segment start angle** and **Segment end angle** parameters affect the time it takes to synchronize the angle measurement, for details refer to Logging the Camshaft Signal Within the Evaluation Segment on page 162.



DS1401 RPCU CAM TPU

Unit | Parameters | Cam Phase Measurement | Advanced

Specify crankshaft wheel by wave table: OFF

Camshaft specification

Number of camshaft pulses: 3

Vector of camshaft start angles [0 ... <720]:

[12 252 492] deg

Vector of camshaft end angles [0 ... <720]:

[48 288 528] deg

Camshaft tolerance [0 ... 180]: 1 deg

Edge polarity: Rising edge

Signal polarity before first transition: Low

Segment specification

Segment start angle [0 ... <720]: 6 deg

Segment end angle [0 ... <720]: 186 deg

With the given crankshaft and camshaft settings, synchronization could take place as follows:

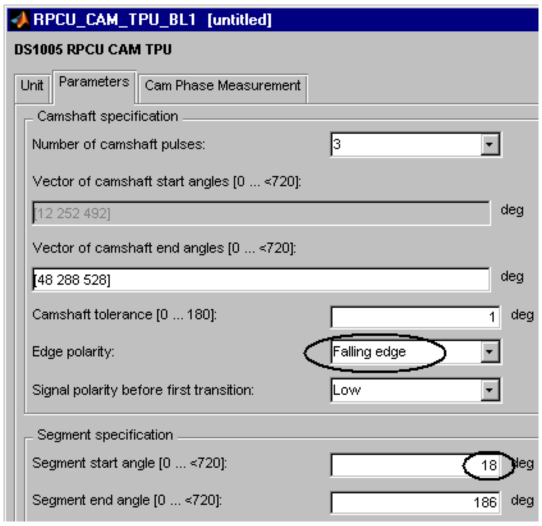
- If the engine starts revolving shortly before the 0° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which is greater than the 6° segment start angle. Thus, the logging of the camshaft signal is not started and synchronization is not possible. However, the angle counter is still running. The 6° segment start angle is reached again after about two engine revolutions (18°+59*12°=726°=6°). Synchronization is finished after about 900°, determined by the segment end angle.
- If the engine starts revolving shortly before the 360° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which is greater than the 6° segment start angle. Thus, the logging of the camshaft signal is not started and synchronization is not possible. However, the angle counter is still running. The 6° segment start angle is reached again after about two engine revolutions (726°=6°). Synchronization is finished after about 900°, determined by the segment end angle.

An improved implementation

In the RPCU_CAM_TPU_BLx block dialog, the Edge polarity is changed to "Falling edge", and the segment start angle is changed to 18°. All the other settings remain unchanged:


RPCU_CAM_TPU_BL1 [untitled]

DS1005 RPCU CAM TPU

Parameters

Camshaft specification

Number of camshaft pulses: 3

Vector of camshaft start angles [0 ... <720]: [12 252 492] deg

Vector of camshaft end angles [0 ... <720]: [48 288 528] deg

Camshaft tolerance [0 ... 180]: 1 deg

Edge polarity: Falling edge

Signal polarity before first transition: Low

Segment specification

Segment start angle [0 ... <720]: 18 deg

Segment end angle [0 ... <720]: 186 deg

With the given settings synchronization can take place as follows:

- If the engine starts revolving shortly before the 0° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which equals the 18° segment start angle. Thus, the logging of the camshaft signal is started. A falling edge at 48° is logged. This matches the marker specification (falling edge at 48°). After the segment end angle (186°) is reached, synchronization is successfully finished.

- If the engine starts revolving shortly before the 360° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which equals the 18° segment start angle. Thus, the logging of the camshaft signal is started. A falling edge at 168° is logged. This matches the marker specification (falling edge at 528°=168°+360° due to segmentation, refer to Updating the Angle Counter with the Camshaft Marker Specification on page 164). After the segment end angle (186°) is reached, synchronization is successfully finished.

Tip

Synchronization can be efficient in the following case:

```
Start angle ≤ Segment start angle ≤ Min relevant camshaft angle
```

In the above example: 18° ≤ 18° ≤ 48°. It was helpful to change the camshaft edge polarity to "Falling edge".

Note

The Segment start angle must always read as follows:

```
Segment start angle = Start angle + N * Period measurement angle
```

Related topics

HowTos

Synchronization Without Crankshaft Gap

Objective

This section provides basic information on synchronizing the angle counter (TCR2) with the camshaft signal if the crankshaft wheel does not have a gap.

Synchronization process

Synchronizing the angle measurement is an internal process and starts automatically after the engine has started. It comprises the following steps:

1. As soon as a crankshaft signal edge is detected, the angle counter is set to the user-defined start value (default: 0) and angle measurement starts with a preliminary 0° position.

   As soon as a crankshaft signal edge has been detected, the following feature is active:

   - Measurement of the engine's speed

2. As soon as a camshaft signal edge is detected, the angle counter is reset to the angle value of the camshaft specification that corresponds to the signal edge.

   As soon as angle measurement is synchronized, the following features are active:

   - Measurement of the engine's position
   - Generation of injection and ignition pulses

Note

If the crankshaft wheel does not have a gap, synchronization does not make use of the camshaft evaluation segment. For a comparison, refer to Synchronization With Crankshaft Gap(s) on page 159.

Only one marker per camshaft wheel

If the crankshaft wheel does not have a gap, the camshaft wheel(s) used for synchronization must not have more than one camshaft marker. If they had two or more, synchronization would not be possible as a detected edge could not be assigned to a specific camshaft marker.

Advantage of multiple camshaft wheels

Using more than one camshaft wheel accelerates synchronization, as it takes less time to detect a camshaft signal edge.

Note

Using multiple camshaft wheels, each with one marker, is similar to using a camshaft wheel with multiple markers (crankshaft wheel has a gap). Refer to Example of the Benefit of a Camshaft Wheel with Multiple Camshaft Markers on page 179.

Where to go from here

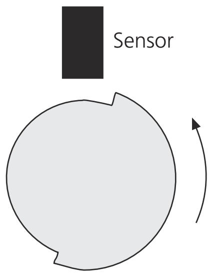
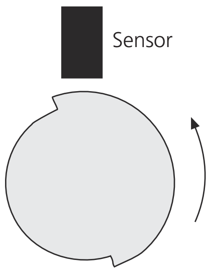
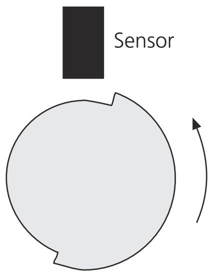
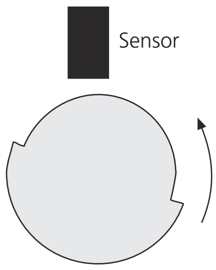
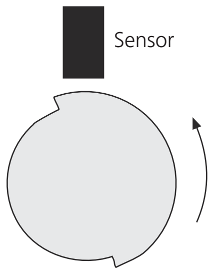
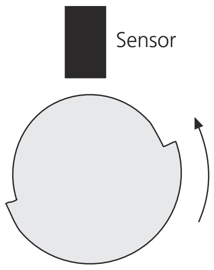
Information in this section

Information in other sections

Example of Synchronization with Multiple Camshaft Wheels

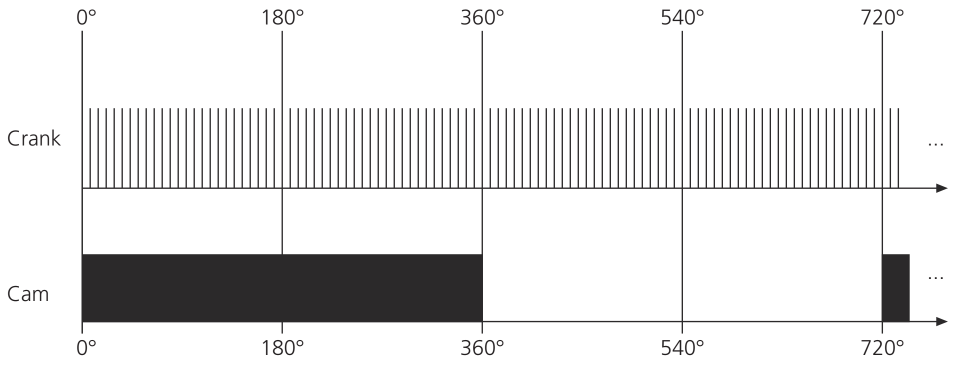
Camshaft wheels

Suppose you have an engine with one camshaft wheel, and another engine with two camshaft wheels. The crankshaft wheels have no gaps. Each camshaft marker comprises an angle range of 360°. For the engine with the two camshaft wheels, the wheels are arranged orthogonally. The polarity of the camshaft signal is set to “Either edges”.

Number of Camshaft Wheels	**Time Required for Synchronization:**	
	Best Case	**Worst Case**
One	 immediately	 after about a half camshaft revolution
Two	  immediately (signal 1)	  after about a fourth camshaft revolution (signal 2)

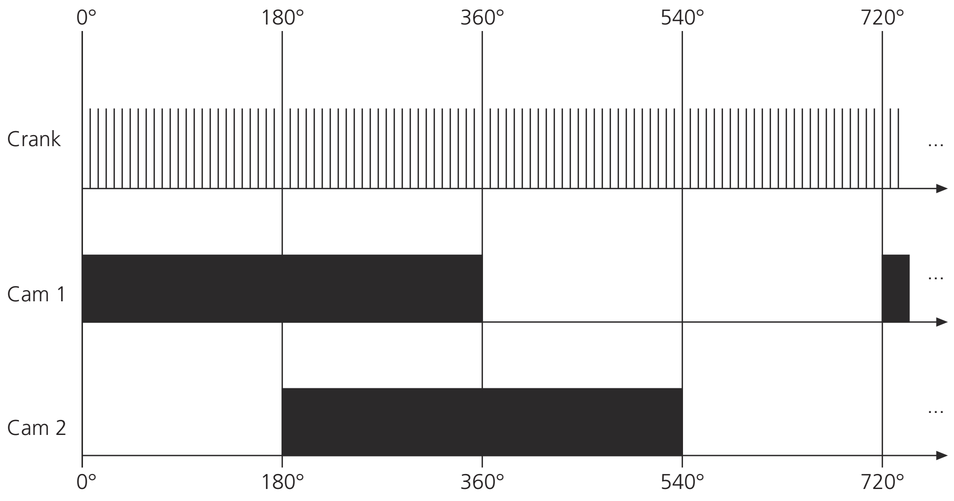
Signals of one camshaft wheel

The following crankshaft and camshaft signals are generated in the case of one camshaft wheel:



Signals of two camshaft wheels

The following crankshaft and camshaft signals are generated in the case of two camshaft wheels:



Synchronization time

In the best case, for both scenarios, initialization is achieved immediately. In the worst case, however, the engine with one camshaft wheel is initialized after about one crankshaft revolution (half a camshaft revolution), but the engine with the two camshaft wheels is initialized after about half a crankshaft revolution (fourth camshaft revolution).

Related topics

Examples

Example of the Benefit of a Camshaft Wheel with Multiple Camshaft Markers.............................. 179

Synchronization With Camshaft Phase Shift

Objective

This section provides basic information on synchronizing the angle counter (TCR2) with the camshaft signal if the camshaft is shifted.

Definition of a camshaft phase shift

Camshaft phase shift means an angular displacement of a camshaft relative to the coupled crankshaft, triggered by an external mechanism. For details, refer to Basics on Camshaft Phase Shift Measurement on page 234.

Camshaft status and synchronization

There are three modes for angle measurement synchronization if camshaft phase measurement is activated (“Lock position”, “Always”, and “Ignore”). The modes concern the position of the camshaft when synchronization starts. The first two modes are described in detail:

- The camshaft must not be shifted (“Lock position” - default). Refer to Synchronization Only in Initial Camshaft Position on page 193
- The camshaft can be shifted (“Always”). Refer to Synchronization in Any Camshaft Position on page 194

Refer to Enable phase measurement on the CAM Phase Measurement Page (RPCU_CAM_TPU_BLx).

Where to go from here

Information in this section

Information in other sections

Synchronization Only in Initial Camshaft Position

Objective

This section provides basic information on allowing synchronization of the angle counter (TCR2) with the camshaft signal only if the camshaft is in initial position (not shifted).

Possible use cases

Before synchronization of the angle measurement is due to start, the camshaft can basically have one of the following start positions (refer to Camshaft start position (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)):

- The camshaft is already in initial position, and initialization can start immediately.
- The camshaft is shifted, and has to be shifted back into initial position before initialization can start.

Note

If you set the camshaft start position to "Shifted", initialization will always work. Thus, this option appears preferable. However, you might waste time if the camshaft is actually in initial position. This is because it takes some time to check the phase shift status of the camshaft. This check is not performed if you select "Init".

Synchronization as usual

Synchronization of the angle measurement is then performed as usual. Refer to Synchronization With Crankshaft Gap(s) on page 159 and Synchronization Without Crankshaft Gap on page 188.

Monitoring the camshaft status

After synchronization of the angle measurement is achieved, the information about the camshaft status (shifted: yes/no), has to be fed to the inport of the RPCU_CAM_TPU_BLx block as a Boolean signal. If synchronization is lost, this status information is used for re-synchronization.

Related topics

Basics

Synchronization in Any Camshaft Position

Objective

This section provides basic information on allowing synchronization of the angle counter (TCR2) with the camshaft signal regardless of whether the camshaft is in initial position or shifted.

Camshaft tolerance parameter

The camshaft shift must not exceed a range that you specify via the Camshaft tolerance parameter. Refer to RPCU_CAM_TPU_BLx. Otherwise, synchronizing the angle measurement is impossible, or erroneous.

Note

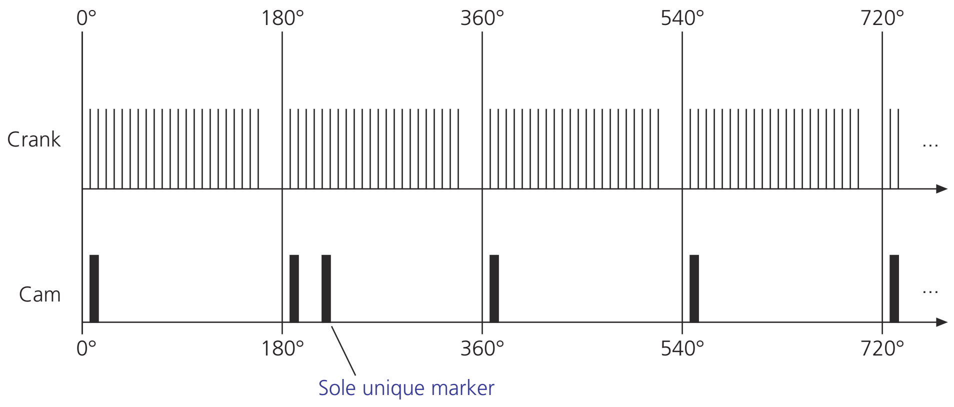
A camshaft phase shift works in one direction. However, the tolerance parameter works in two directions: - and +. Refer also to Example of Synchronization With Camshaft Phase Shift on page 196.

Crankshaft gap required

Synchronizing the angle measurement when the camshaft is shifted is only sensible if the crankshaft wheel has at least one gap. Because, without a crankshaft gap, angle measurement can only be as precise as the camshaft tolerance allows.

Multiple camshaft markers

With reference to the engine's start position and the camshaft tolerance parameter, the specified angle values of multiple camshaft markers must not overlap (see illustration below) as this may lead to erroneous synchronization. The camshaft evaluation segment must cover at least one unique camshaft marker.



Synchronization as usual

Synchronization of the angle measurement is then performed as usual. Refer to Synchronization With Crankshaft Gap(s) on page 159 and Synchronization Without Crankshaft Gap on page 188.

Related topics

Examples

Synchronization and Ignored Camshaft Wheels

Objective

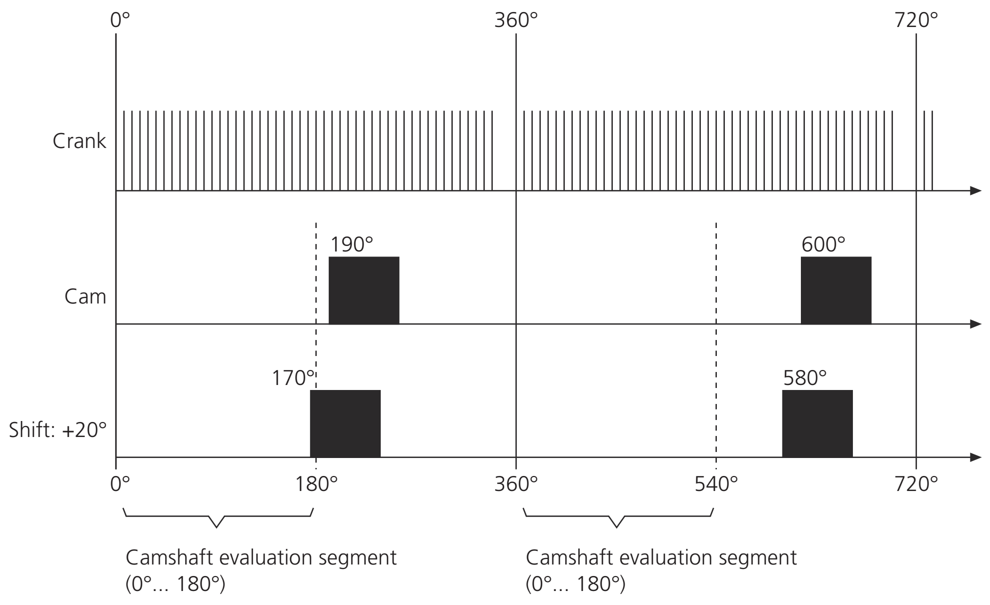
This section provides basic information on allowing the signal of a camshaft wheel to be used only for the camshaft phase shift measurement function but not for synchronizing the angle counter (TCR2) with the camshaft signal.

Camshaft phase shift measurement

After synchronization is achieved, the edges of the camshaft signal are detected, and the related angle values are compared to the angle values of the camshaft marker specification. The difference between each angle value pair is returned as the current camshaft phase shift.

Risk of invalidity

A problem arises if an ignored camshaft is shifted too much during synchronization. An edge of the camshaft signal of the ignored camshaft wheel might be shifted into, or out of, the camshaft evaluation segment.


0°
360°
720°
Crank
Cam
190°
600°
Shift: +20°
170°
580°
0°
180°
360°
540°
720°
Camshaft evaluation segment (0°... 180°)
Camshaft evaluation segment (0°... 180°)

In consequence, comparing the detected edges and the camshaft marker specification does not work, as incompatible values are compared. The first angle value measured is always compared with the first angle value specified, and so on. As shown in the illustration above, for example, 580° is compared with 190° (phase shift: -390°), instead of comparing 170° with 190° (phase shift: +20°).

This incompatibility continues for all subsequent measurements and makes the whole camshaft phase shift measurement invalid.

Thus, a camshaft wheel that is ignored during synchronization can only be shifted within specific limits during synchronization (no edge must be shifted into, or out of, the camshaft evaluation segment). The easiest way to counter this limitation is to have the camshaft in initial position during synchronization.

Note

The RapidPro system does not check whether there is camshaft phase shift during synchronization. You have to ensure that the camshaft wheel is in initial position.

Related topics

HowTos

How to Exclude a Camshaft Wheel from Synchronization..213

Example of Synchronization With Camshaft Phase Shift

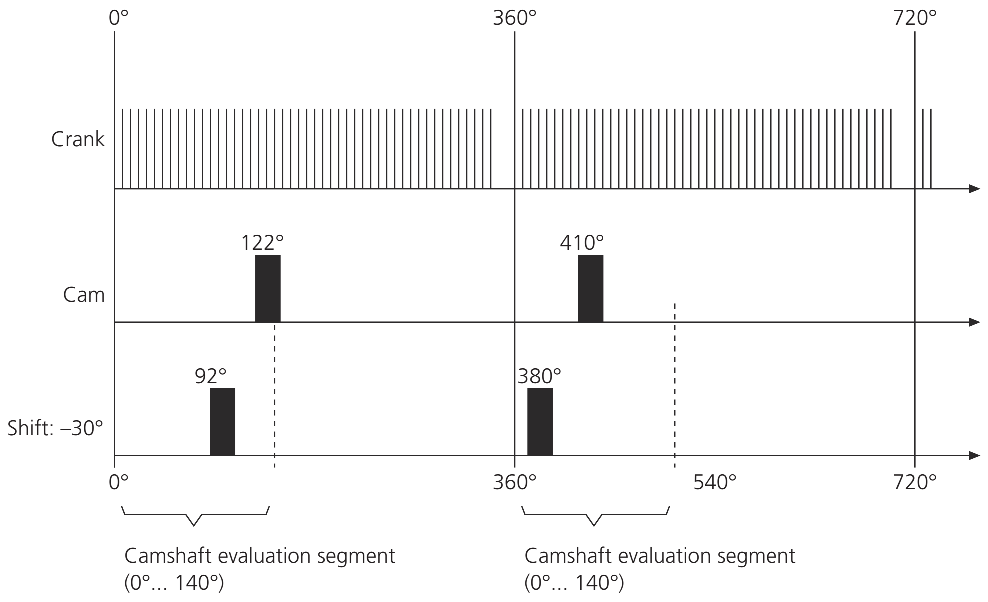
Synchronization example

This example shows how the angle measurement can be synchronized even though the camshaft is shifted.

Suppose the crankshaft and camshaft parameters are as follows:

Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth
Camshaft	Markers: ▪ #1: 122° … 154° (1st crankshaft revolution) ▪ #2: 410° … 436° (2st crankshaft revolution) Edge polarity: ▪ “Rising edges” Camshaft evaluation segment: ▪ 0° … 140° Camshaft phase shift at synchronization: ▪ -30° Camshaft tolerance: ▪ ±35°

The following crankshaft and camshaft signals are generated:


0°
360°
720°
Crank
Cam
122°
410°
Shift: –30°
92°
380°
0°
360°
540°
720°
Camshaft evaluation segment (0°... 140°)
Camshaft evaluation segment (0°... 140°)

Note

If a camshaft phase shift with the opposite orientation, for example, +30°, was possible, the camshaft evaluation segment would have to be larger.

Marker #1

Suppose the engine is started in such a position that camshaft marker #1 is detected first (specified angle value of the rising edge is 122°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
92°	–

Comparing the logged angle values with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
122° (±35°) 410° (±35°)	92°	Yes	Yes

The logged angle values match the specified angle values. Synchronization is achieved after 100°, as this is the size of the camshaft evaluation segment.

Marker #2

Suppose the engine is started in such a position that camshaft marker #2 is detected first (specified angle value of the rising edge is 410°). The camshaft angle values are logged as follows:

Logged Edges	
Rising	**Falling**
20°	–

Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
122° (±35°) 410° (±35°)	20°	No	No

The logged angle values do not match the specified angle values. Synchronization is not achieved.

Subsequently, the internal algorithm corrects the logged angle values (+360°). Comparing the logged camshaft angle value with the specified camshaft angle values now returns the following:

Specified Angles	Logged Angles	Match	Synch.
122° (±35°) 410° (±35°)	380°	Yes	Yes

The logged angle values match the specified angle values. Synchronization is achieved after 100°, as this is the size of the camshaft evaluation segment. The angle counter (TCR2) is updated (+360°).

Conclusion from this example

The camshaft tolerance can be used to synchronize despite a camshaft phase shift. The tolerance must be larger than the camshaft phase shift. However, the camshaft markers must be unambiguous, despite the tolerance.

Note

A camshaft phase shift commonly has a certain orientation (– or +). The camshaft tolerance, however, covers both orientations (±). Thus, half the tolerance is actually not made use of.

Tip

If a too large camshaft tolerance makes the specification of multiple camshaft markers ambiguous, and you know the maximum possible camshaft phase shift and its orientation at synchronization time, you can use the following workaround:

- In the RPCU_CAM_TPU_BLx:
  - Reduce the camshaft tolerance
  - Shift the angle values of the camshaft marker specification instead
- In the RTI model:
  - Reshift the results of the camshaft phase shift measurement

Related topics

Basics

Setting Up Crankshaft and Camshaft Signal Measurement

Objective

Before you can use the RapidPro engine control features, you first have to set up the measurement of the crankshaft and camshaft signals. This task requires a knowledge of the required input signals and the angle counter synchronization.

Note

Synchronization requires a speed of 40 ... 20000 rpm.

Where to go from here

Information in this section

Information in other sections

Basics on Crankshaft Wheel Specification

Objective

You can specify a crankshaft wheel either via parameters or via a wavetable. The kind of specification influences the crankshaft signal processing.

Note

The wavetable specification approach requires a RapidPro Control Unit with MPC565 not older than board revision DS1601-12.

Specification opportunities

The specification of a crankshaft wheel defines the location of the rising and falling tooth edges in relation to the sensor signal. You can specify a crankshaft wheel as follows:

Wheel Type	Specification Is Possible Via ...[1]
Symmetric	Parameters: ▪ Number of crankshaft teeth ▪ Missing teeth per gap ▪ Number of gaps — or — Wavetable: ▪ Wavetable file
Asymmetric	Parameters: ▪ Not possible! Wavetable: ▪ Wavetable file

[1] Refer to Parameters Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Wheel wavetable

A wavetable comprises 7200 data points (corresponding to an angular resolution of 0.05°). The data points must read 1 (= tooth) or 0 (= no tooth). A 0-1 transition represents a rising tooth edge, and a 1-0 transition a falling tooth edge.

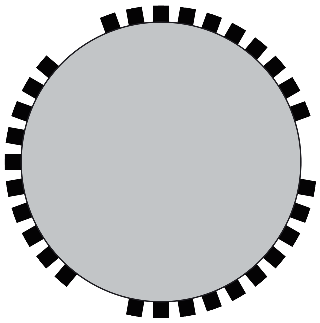
Tip

Specifying a crank wheel via wavetable is advantageous because:
- The gaps and teeth of a crankshaft wheel can have arbitrary length.
- Asymmetric crankshaft wheels can be used.
- No need to specify a camshaft evaluation segment.
- Synchronization can be achieved faster.
- Speed measurement is more precise if speed changes abruptly, for example, at a gap.
- Speed measurement results can be smoothed.
- Reverse crankshaft rotations can be handled.

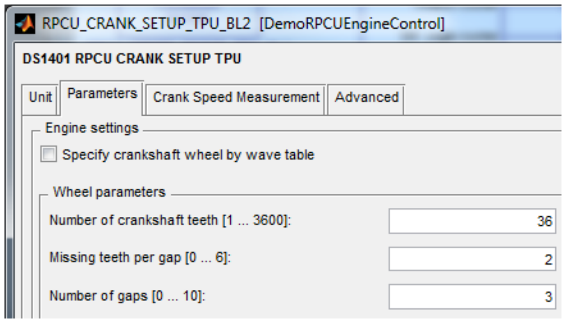
Limitation With this version of the RTI RapidPro Control Unit Blockset, you cannot specify crankshaft wheels without a gap by using a wavetable, but only by setting parameters.

Specification example

Suppose you want to specify a 36-3-2 crankshaft wheel. This crankshaft wheel is symmetric. It has 36 teeth, 3 gaps, and 2 missing teeth per gap.



Wheel parameters As the wheel is symmetric, you can specify wheel parameters, for example:



RPCU_CRANK_SETUP_TPU_BL2 [DemoRPCUEngineControl]

DS1401 RPCU CRANK SETUP TPU

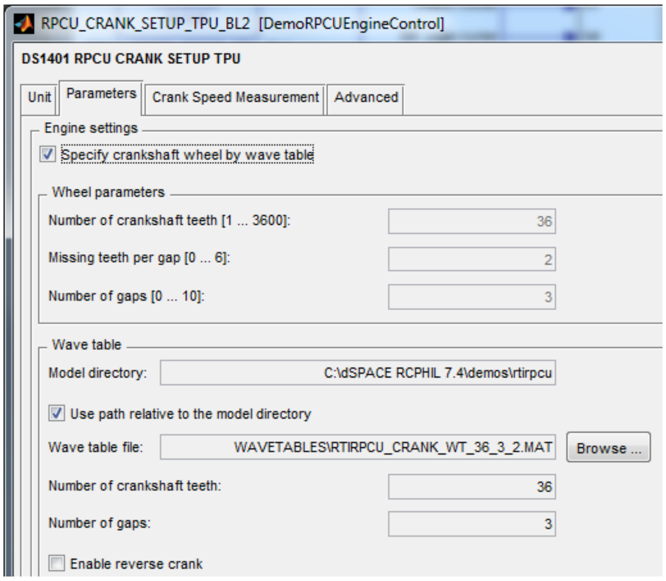
Engine settings

☐ Specify crankshaft wheel by wave table

Wheel parameters

Number of crankshaft teeth [1 ... 3600]:	36
Missing teeth per gap [0 ... 6]:	2
Number of gaps [0 ... 10]:	3

Wheel wavetable All kinds of crankshaft wheels can be specified by referencing a wavetable, for example:


RPCU_CRANK_SETUP_TPU_BL2 [DemoRPCUEngineControl]

DS1401 RPCU CRANK SETUP TPU

Unit | Parameters | Crank Speed Measurement | Advanced

Engine settings

[✓] Specify crankshaft wheel by wave table

Wheel parameters

Number of crankshaft teeth [1 ... 3600]:	36
Missing teeth per gap [0 ... 6]:	2
Number of gaps [0 ... 10]:	3

Wave table

Model directory: C:\dSPACE RCPHIL 7.4\demos\rtirpcu

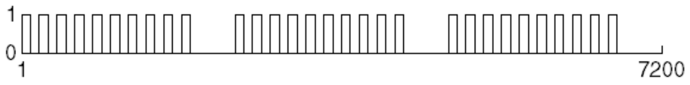
[✓] Use path relative to the model directory

Wave table file: WAVETABLES\RTIRPCU_CRANK_WT_36_3_2.MAT Browse ...

Number of crankshaft teeth:	36
Number of gaps:	3

[ ] Enable reverse crank

The following illustration is a plot of a wavetable (7200 data points, 0 or 1) that defines a 36-3-2 crankshaft wheel.



Note

The first data point relates to the 0°-position, if the Start angle parameter is set to 0°.

A wavetable consists of 7200 single bit values. 1 means tooth, 0 means no tooth. The angular resolution of the single bit values is 0.05° (360°/7200).

Ready-to-use wavetables

For a selection of commonly used symmetric crankshaft wheels, your RapidPro software provides suitable wavetables as MAT files in `%DSPACE_ROOT%\Demos\RTIRPCU\Wavetables`.

- rpcu_crank_wt_36_2_2.mat
- rpcu_crank_wt_36_3_2.mat
- rpcu_crank_wt_60_1_2.mat
- rpcu_crank_wt_60_2_2.mat

Note

For the wavetables that come with the dSPACE software, the first data point relates to the first rising edge after a gap.

Creating wavetables

To create your own wavetables you can use the `rtirpcu_crank_wt_create.m` M script, which resides in the same folder.

Related topics

Basics

How to Prepare Crankshaft Signal Processing

Objective

To set up crankshaft signal evaluation, which is an essential precondition for performing engine control, you must first perform a few preparation steps.

Preconditions

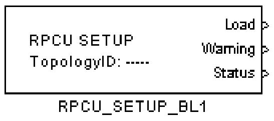
Before you start preparing crankshaft signal processing, the following preconditions must be met:

- A valid RapidPro hardware topology file (`*.hwt`) is available.
- The RapidPro Library: rtirpculib is opened.
- The controller model is opened.

Method

To prepare crankshaft signal processing

1 From the RapidPro Library: rtirpculib, copy an RPCU_SETUP_BLx block to the model.



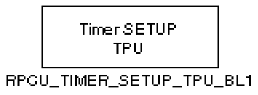
2 Open the block, and select the Unit page.

3 In the Board/Module number list, choose the desired board/module number.

4 Browse for a valid RapidPro hardware topology file (`*.hwt`) and open it.

The topology ID of the RapidPro hardware used is displayed in the TopologyID field.

5 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_TIMER_SETUP_TPU_BLx block to the model.



6 Open the block, and select the Unit page.

7 In the **Board/Module number** list, choose the desired board/module number.

8 In the **ECU channel** list, choose the desired ECU channel number. (Only if you work with a DS1007 modular system)

9 Select the **Parameters** page.

The settings of the time counter registers (TCR1 and TCR2) of the three TPU units (A, B, and C) are shown.

10 Select the desired **Use TPU [...] timer 2 for engine control** checkbox(es) for all TPU units used for engine control.

A **TPU [...] timer 2 resolution** list that belongs to a selected checkbox is disabled, as a timer 2 is not used as a timer but as an angle counter (TCR2).

11 Select the timer resolution of the remaining timers.

Result

You can start setting up the crankshaft signal processing.

Related topics

HowTos

How to Set Up Crankshaft Signal Processing.. 205

How to Set Up Crankshaft Signal Processing

Objective

If you want to set up crankshaft signal evaluation, which is an essential precondition for performing engine control, you have to perform the following steps.

Speed measurement

Evaluating the crankshaft signal allows you to measure the engine speed.

Preconditions

Note

The number of crankshaft teeth must be an integer divisor of 360, for example, 30 or 36.

Before you start setting up crankshaft signal processing, the following preconditions must be met:

- The crankshaft signal is connected to an SC module mounted on the RapidPro Control Unit, not on a RapidPro SC Unit.
- You have prepared the setup, refer to How to Prepare Crankshaft Signal Processing on page 204.

Method

To set up crankshaft signal processing

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_CRANK_SETUP_TPU_BLx block to the model

Crankshaft
SETUP
RPCU_CRANK_SETUP_TPU_BL1

2 Open the block, and select the Unit page.

The topology ID of the RapidPro hardware used is displayed in the TopologyID field.

3 In the Board/Module number list, choose the desired board/module number.

4 From the inport selection list, choose an input channel for crankshaft signal capture.

The Ch01 and Ch02 channels of the related TPU are used internally and thus not available while engine control is performed.

5 Select the Parameters page.

6 If you want to reference a MAT wavetable file, select Specify crankshaft wheel by wavetable.

7 Specify the crankshaft wheel.

If a wavetable is referenced, you can detect a reverse crankshaft rotation, refer to the Enable reverse crank option.

8 Select the Crank speed measurement page.

9 Specify the settings for measuring of the crankshaft speed.

10 Select the Advanced page.

11 Specify the settings for generating of periodic angle-based pulses. Configure the speed calculation suppression (available only, if the crankshaft wheel is specified by parameters, not by a wavetable).

Result

Crankshaft signal processing is set up.

Note

Setting up crankshaft signal processing is an essential precondition for performing engine control. It allows you to use the engine control feature that is not based on angle measurement, that is, engine speed measurement. The camshaft signal processing has to be set up for all the engine control features that are based on angle measurement, for example, measuring the engine position and camshaft phase shift.

Related topics

Basics

HowTos

References

How to Set Up Camshaft Signal Processing

Objective

To set up camshaft signal evaluation, which is essential for using engine control features based on the engine's position, you have to perform the following steps.

Full access to engine control features

After camshaft and crankshaft signal processing have been set up, all possible set up work has been done. This allows you to implement all RapidPro engine control features, for example, generating injection and ignition pulses.

One RTI block per wheel

Each camshaft wheel requires its own RPCU_CAM_TPU_BLx block, in other words, one RPCU_CAM_TPU_BLx block specifies one camshaft wheel.

Preconditions

Before you start setting up camshaft signal processing, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is open.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.

Method

To set up camshaft signal processing

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_CAM_TPU_BLx block to the model.

Camshaft
RPCU_CAM_TPU_BL1

2 Open the block, and select the Unit page.

3 Configure the settings of the **Unit** page to match their counterpart settings on the **Unit** page of the RPCU_CRANK_SETUP_TPU_BLx block. If the RPCU_CRANK_TPU_BLx block uses
   - the same TPU unit, TPU channels Ch01 and Ch02 are used internally and thus not available during engine control is performed.
   - another TPU unit, TPU channel Ch01 is used internally and thus not available during engine control is performed.

4 Select the **Parameters** page.

   The specifications of the camshaft wheel and the camshaft evaluation segment are shown.

5 Specify the camshaft parameters.

   Note

   - If the crankshaft wheel is specified via wavetable, a camshaft evaluation segment does not need to be specified.
   - If the crankshaft wheel is specified via parameters, the segment specification is important for synchronizing angle measurement. Refer to Logging the Camshaft Signal Within the Evaluation Segment on page 162 and Synchronization in Any Camshaft Position on page 194.

6 Select the **Cam phase measurement** page.

   The parameters that manage a camshaft phase shift are shown.

7 Specify the camshaft phase shift parameters.

8 Select the **Advanced** page if you want to force synchronization without camshaft signal edge detection.

   Note

   If the crankshaft wheel is specified via wavetable, this feature is always active (though the **Force synchronization without camshaft signal edge detection** checkbox is disabled and cleared).

Result Camshaft signal processing is set up.

Related topics

Basics

HowTos

References

How to Set Up Engine Offsets

Objective

If you do not want the 0° position to match the default 0° position, you should specify an offset.

Default 0° position

The default 0° position depends on the crankshaft wheel specification as follows:

- Specification via wavetable:

  The 0° position matches the first wavetable data point.

  Note

  For the wavetables that come with the dSPACE software, the first data point relates to the first rising crankshaft signal edge after a gap.

- Specification via parameters:

  The 0° position matches the first relevant crankshaft signal edge after a gap (rising or falling edge, which depends on the **Edge polarity**).

Impact of the crankshaft wheel specification

The specification of the crankshaft wheel (wavetable versus parameters) affects how an engine offset is set up:

1. Always, follow the instructions of Method 1.
2. If the crankshaft wheel is specified via parameters, also follow the instructions of Method 2.

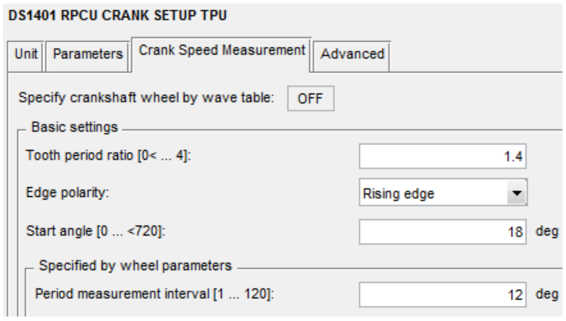
Method 1

To set up an engine offset

1 Calculate the offset [0°...719.999°]. It is always a positive number (e.g., -20° is not allowed, in this case the offset is 700°).

For example, if you have a 60 – 2 wheel (60 teeth, one gap with two missing teeth), and the engine's 0° position is at the rising edge of the last tooth *before* the gap, the offset equals 18°.

2 Open the related RPCU_CRANK_SETUP_TPU_BLx block, and select the Crank Speed Measurement page.



Specify crankshaft wheel by wave table: OFF

Basic settings

Tooth period ratio [0< ... 4]:	1.4
Edge polarity:	Rising edge
Start angle [0 ... <720]:	18 deg

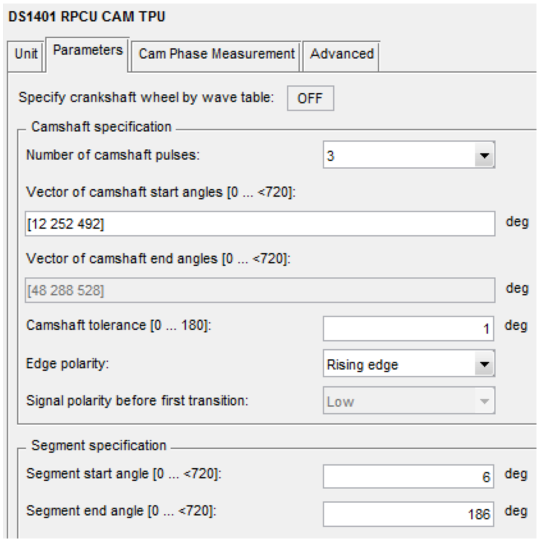
Specified by wheel parameters

Period measurement interval [1 ... 120]:	12 deg

3 Change the Start angle as follows (default: 0°):

Start angle_Offset = Start angle + Offset

4 Open the related RPCU_CAM_TPU_BLx block, and select the Parameters page.



Specify crankshaft wheel by wave table: OFF

Camshaft specification

Number of camshaft pulses:	3
Vector of camshaft start angles [0 ... <720]:	[12 252 492] deg
Vector of camshaft end angles [0 ... <720]:	[48 288 528] deg
Camshaft tolerance [0 ... 180]:	1 deg
Edge polarity:	Rising edge
Signal polarity before first transition:	Low

Segment specification

Segment start angle [0 ... <720]:	6 deg
Segment end angle [0 ... <720]:	186 deg

5 Change the Vector of camshaft start angles and Vector of camshaft end angles as follows:

Camshaft start angle_Offset = Camshaft start angle + Offset

Camshaft end angle_Offset = Camshaft end angle + Offset

Note

Check whether the modified angle values are still strictly monotonic increasing, otherwise resort them.

6 Check whether the Signal polarity before first transition setting (Low/High) is still correct, as the 0° position has been shifted.

> **Note**
>
> The Signal polarity before first transition parameter is only evaluated if the Edge polarity parameter is set to "Either edge".

Method 2

To set up an engine offset (only for crankshaft wheel parameters)

> **Note**
>
> The following steps are additionally required if the crankshaft wheel is specified via parameters.

1 Change the Segment start angle as follows:

Segment start angle_Offset = Start angle_Offset + N * Period measurement angle

N is an integer multiple.

To achieve quick synchronization, the segment start angle should read as follows:

Start angle ≤ Segment start angle ≤ Minimum relevant camshaft angle

> **Tip**
>
> It might also be useful to change the Period measurement interval instead of changing the Segment start angle.

2 Change the Segment end angle as follows:

Maximum relevant camshaft angle ≤ Segment end angle ≤ (360°÷No. of gaps)

The segment end angle should read as follows:

- As small as possible, as otherwise synchronization takes longer than actually necessary.
- Large enough for all the relevant camshaft angle values to be logged, even if the camshaft is shifted.

Result

You have set up an engine offset.

How to Define the Camshaft Position at Synchronization

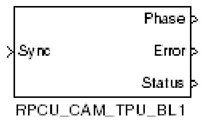
Objective

If you perform camshaft phase shift measurement, you have to specify whether a shifted camshaft can remain shifted or has to be re-shifted before synchronization can start.

Method

To define the camshaft position at synchronization

1 Open the related RPCU_CAM_TPU_BLx block.



2 Select the Cam Phase Measurement page.

3 In the Synchronization mode list, choose:

- "Always", if the camshaft can remain shifted
- "Lock position", if the camshaft has to be shifted back first

Result

The camshaft position at synchronization is defined for the camshaft that belongs to the RPCU_CAM_TPU_BLx block.

Related topics

Basics

HowTos

References

How to Exclude a Camshaft Wheel from Synchronization

Objective

If you perform camshaft phase shift measurement and want to exclude the camshaft from synchronization, you have to perform the following steps.

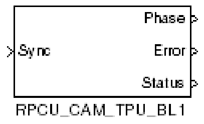
Initial position at synchronization

An excluded camshaft should be in initial position during synchronization, that is, there must be no camshaft phase shift.

Method

To exclude a camshaft wheel from synchronization

1 Open the related RPCU_CAM_TPU_BLx block.



2 Select the Cam Phase Measurement page.

3 In the Synchronization mode list, choose “Ignore”.

Result

The camshaft wheel that belongs to the RPCU_CAM_TPU_BLx block is excluded from synchronization.

Note

Ensure that the ignored camshaft is in initial position before synchronization starts.

Related topics

Basics

Basics on Camshaft Sensor Signals.. 145
Synchronization and Ignored Camshaft Wheels.. 195

HowTos

How to Define the Camshaft Position at Synchronization.. 212
How to Set Up Camshaft Phase Shift Measurement... 236

References

Camshaft Signal Reference Data... 273

Measuring Engine Speed, Position, and Status

Objective

The RapidPro Control Unit provides special input features that you can use for measuring engine speed, position, and status.

Where to go from here

Information in this section

Information in other sections

Basics on Engine Speed, Position, and Status (Crank Parameters)

Where to go from here

Information in this section

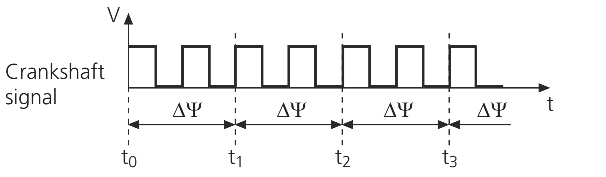
Information in other sections

Basics on Speed Measurement (Crank Parameters)

Speed calculation method

Speed is calculated as revolutions per minute (rpm). The time that the crankshaft takes to rotate over a user-defined angle range is measured.

You have to specify the **Period measurement interval** (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖) parameter. This parameter defines the angle distance after which speed measurement is repeatedly performed (in the illustration below, the crankshaft signal is displayed with a large zoom factor).



$\Delta\Psi$:	Period measurement angle(user-derfined)
t_i:	Time measured by TCR1

Refer to Period measurement interval (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

Range of the speed measurement The possible range of the speed measurement (rpm: revolutions per minute) is as follows:

Rpm_{min}	Rpm_{max}
40	20000

Note

The possible range of the speed measurement does not depend on the TPU prescaler setting. Speed resolution (TCR1), however, does.

Preconditions for the speed measurement

Basically, the speed measurement is derived from the period measurement. In addition, if the crankshaft wheel has a gap, speed measurement is not possible until after a crankshaft gap has been detected. For details of gap detection. Refer to Triggering the Angle Counter via Gap Detection on page 161.

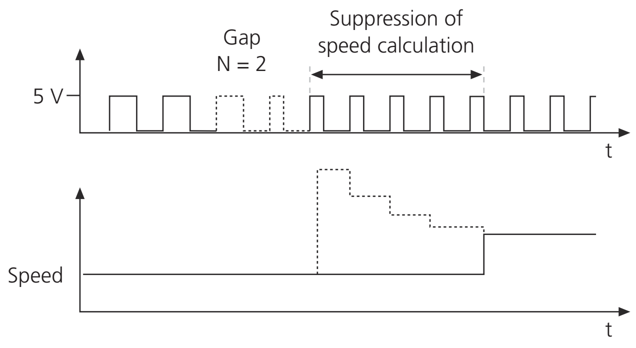
Speed measurement at a gap

The same speed measurement mechanism is applied whenever a gap appears: 0.1° pulses are counted, the elapsed time is measured, and the speed is calculated. What is actually important for you to know is the way the crankshaft signal is transformed into 0.1° pulses at a gap. Refer to Processing the Crankshaft Signal on page 147.

Suppression of speed calculation after a gap

You can suppress the speed calculation during a specific angle range succeeding a gap of the crankshaft wheel (refer to Advanced Page

(RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)). The speed is not calculated until a user-defined number of teeth has passed by (1 ... 4). The last calculated speed value is used during the suppression phase, instead. In the illustration below, speed calculation is suppressed for four teeth.



This feature is helpful to the following uses case:

- The period measurement angle equals the smallest possible value, that is:

  `Period measurement angle = 360° / No. of crankshaft teeth`
- Crankshaft rotation speed increases too strongly during the gap, for example, due to ignition. For details on possible acceleration rates, refer to Limitation of the Speed Measurement (Crank Parameters) on page 217.
- In consequence, the calculated speed curve may have a peak overshoot after the gap.

Tip

To avoid peak overshoots of the calculated speed values you could also increase the **Period measurement interval** (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

Related topics

Basics

HowTos

References

Crank Speed Measurement Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

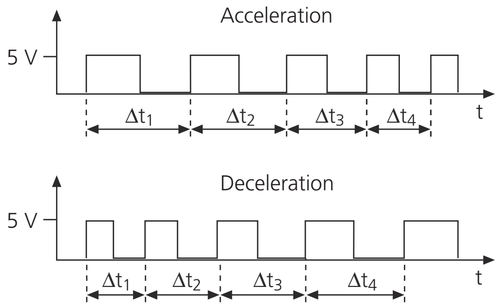
Limitation of the Speed Measurement (Crank Parameters)

Objective

This section provides basic information on the restriction of the RapidPro speed measurement algorithm.

Acceleration and deceleration

Changes in speed are limited. The user-defined **Ratio** parameter determines the factor that the time (Δt_{i-1}) is allowed to change by the next period (Δt_i). The **Ratio** parameter is used to detect invalid changes in speed caused by any kind of defect such as a broken wire or signal distortion. (The illustration below has a large zoom factor).



Restrictions Acceleration and deceleration must meet the following restrictions:

- Acceleration:

  $\Delta t_i \geq \Delta t_{i-1}$ / Ratio
- Deceleration:

  $\Delta t_i \leq \Delta t_{i-1}$ * Ratio

Note

Typically, the **Ratio** parameter fits the range 1.4 … 1.9.

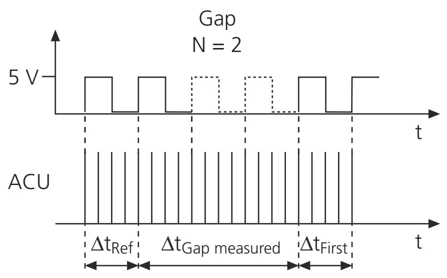
Consequences in case of violation

Violating the limitation specified by the Ratio parameter has the following results:

1. An error is generated.
2. Synchronization is restarted.

Acceleration and deceleration at a crankshaft gap

At a crankshaft gap, the ratio parameter is applied analogously as for teeth. However, the speed is assumed to be constant during the gap. The time $\Delta t_{\text{Gap measured}}$ is measured from the edge of the tooth preceding the gap to the edge of the tooth succeeding the gap, as illustrated below (with a large zoom factor). Δt_{Ref} refers to the second last tooth before the gap, Δt_{First} refers to the first tooth after the gap.



N: Number of missing teeth

Note

The **Ratio** parameter must not be too large, it must always be smaller than 1+N.
A gap cannot be detected if the first edge after the gap is found *within* the time range defined by the **Ratio** parameter. Thus, ensure that the following applies:
$\Delta t_{Gap\ measured} > \Delta t_{Ref} * \text{Ratio}$

Acceleration and deceleration Acceleration and deceleration at a gap must meet the following restrictions:

- Acceleration:

  $\Delta t_{Gap\ measured} / (1+N) \geq \Delta t_{Ref} / \text{Ratio}$

  $\Delta t_{First} \geq [\Delta t_{Gap\ measured} / (1+N)] / \text{Ratio}$

- Deceleration:

  $\Delta t_{Gap\ measured} / (1+N) \leq \Delta t_{Ref} * \text{Ratio}$

  $\Delta t_{First} \leq [\Delta t_{Gap\ measured} / (1+N)] * \text{Ratio}$

Related topics

Basics

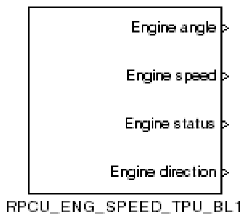
Basics on the Engine Status (Crank Parameters)

Objective

You can measure the engine behavior (position, speed, status) and monitor the synchronization status.

RTI blocks

You can use the RPCU_ENG_SPEED_TPU_BLx block to monitor the engine angle, speed, and status. You can also check the status of the result (new or old).



In addition, you can analyze in detail the occurrence of synchronization errors (refer to How to Analyze Synchronization Errors on page 231).

Engine status

The engine status is indicated by numbers. If the crankshaft wheel is specified via wheel parameters, the **Engine status** port can output these numbers:

Status	Description
0	Crankshaft angle measurement and period measurement have not started. Crankshaft and camshaft signals are not synchronized.
1	Speed measurement and evaluation of the camshaft signal have started.
2	Synchronization algorithm has started. Logging of the camshaft signal has finished.
3	Crankshaft angle measurement is being adjusted (update of TCR2).
4	Crankshaft and camshaft signals are synchronized. Generation of injection and ignition pulses has started.
5	Crankshaft and camshaft signals are synchronized. Injection and ignition pulses are being generated. Angle values are being returned.
6	(Only possible if no camshaft signal exists) Period measurement has started.

Related topics

Basics

HowTos

References

dsrpcu_crank_read (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
RPCU_ENG_SPEED_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Basics on Engine Speed, Position, and Status (Crank Wavetable)

Where to go from here

Information in this section

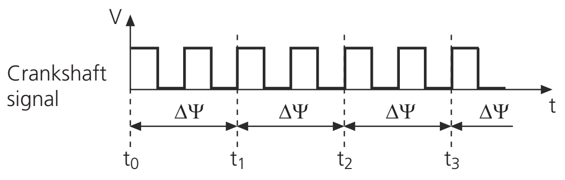
Information in other sections

Basics on Speed Measurement (Crank Wavetable)

Overview

The time that the crankshaft takes to rotate over a user-defined angle range $\Delta\psi$ is measured. Speed is calculated as:

$\text{Speed}(t_i) = \Delta\psi / (t_i - t_{i-1})$



$\Delta\Psi$:	Period measurement angle(user-derfined)
t_i:	Time measured by TCR1

$\Delta\psi$ is based on the ACU angle measurement (crank signal edges).

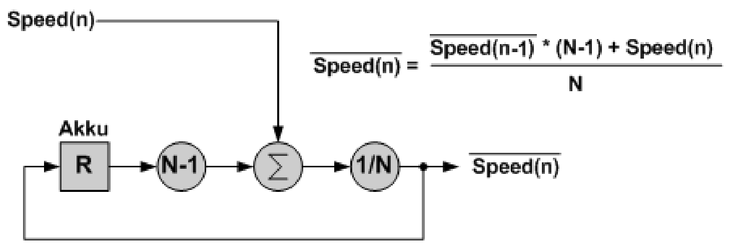
Speed calculation method

This method is applied if the crankshaft wheel is specified via wavetable. You can perform direct or recursive speed measurement.

Direct speed measurement You have to specify the Period measurement interval parameter, which defines the number of signal edges after which speed measurement is repeatedly performed. Adjacent intervals do not overlap. The angular length of adjacent intervals may differ (signal edge to signal edge).

Recursive speed measurement The measured speed value (direct measurement, Period measurement interval = 1) and the last speed result are used.

You have to specify the Output interval and Recursion weighting factor parameters. The first parameter defines the number of signal edges after which the result is repeatedly output. The second parameter defines how the old result contributes to the new result according to the following equation:


Speed(n)
$\overline{\text{Speed(n)}} = \frac{\overline{\text{Speed(n-1)}} * (N-1) + \text{Speed(n)}}{N}$
Akku
R
N-1
∑
1/N
$\overline{\text{Speed(n)}}$

$\overline{\text{Speed(n)}}$	New result
$\overline{\text{Speed(n-1)}}$	Old result
Speed(n)	Measured speed value
N	Recursion weighting factor

Range of the speed measurement

The possible range of the speed measurement (rpm: revolutions per minute) is as follows:

Rpm_{min}	Rpm_{max}
0	10000

Note

Below 5 rpm, speed cannot be measured and is set to 0.

Reverse crankshaft rotation The speed is not measured when the crankshaft rotates in reverse direction. Speed = 0 is returned instead.

Related topics

Basics

Basics on Crankshaft Wheel Specification 201

HowTos

How to Analyze Synchronization Errors 231
How to Measure the Engine Speed 228

References

Crank Speed Measurement Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

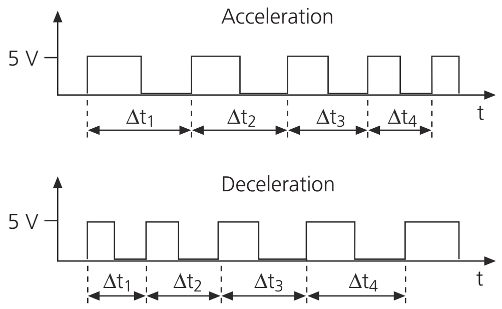
Limitation of the Speed Measurement (Crank Wavetable)

Objective

This section provides basic information on the restriction of the RapidPro speed measurement algorithm.

Acceleration and deceleration

Changes in speed are limited. The user-defined **Ratio** parameter determines the factor that the time (Δt_{i-1}) is allowed to change by the next period (Δt_i). The **Ratio** parameter is used to detect invalid changes in speed caused by any kind of defect such as a broken wire or signal distortion. (The illustration below has a large zoom factor).



Restrictions Acceleration and deceleration must meet the following restrictions (z_i and z_{i-1} are the angular distances between relevant adjacent crank signal edges. They relate to Δt_i and Δt_{i-1}. Note that z_i/z_{i-1} also covers the conditions at a crankshaft gap):

- Acceleration:

  $\Delta t_i \geq [(z_i/z_{i-1}) * \Delta t_{i-1}] / \text{Ratio}$
- Deceleration:

  $\Delta t_i \leq [(z_i/z_{i-1}) * \Delta t_{i-1}] * \text{Ratio}$

Note

Typically, the **Ratio** parameter fits the range 1.4 … 1.9.

Consequences in case of violation

Violating the limitation specified by the Ratio parameter due to too low speed values has the following results:

If synchronization is not yet achieved:

1. Invalid speed measurement

   Crank event error counter: +1
2. Synchronization is restarted.

If synchronization is already achieved:

1. Valid speed measurement

   Crank event error counter: +1

   Note

   The crank event error counter is reset if no crank event error occurs during a 720° interval.

2. Synchronization is restarted if the crank event error counter counts >3.

Related topics

Basics

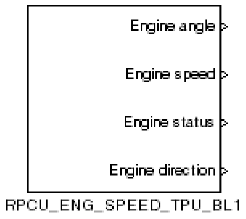
Basics on the Engine Status (Crank Wavetable)

Objective

You can measure the engine behavior (position, speed, status, and direction) and monitor the synchronization status.

RTI blocks

You can use the RPCU_ENG_SPEED_TPU_BLx block to monitor the engine angle, speed, status, and direction. You can also check the status of the result (new or old).



In addition, you can analyze in detail the occurrence of synchronization errors (refer to How to Analyze Synchronization Errors on page 231).

Engine status

The engine status is indicated by numbers. If the crankshaft wheel is specified via wheel parameters, the **Engine status** port can output these numbers:

Status	Description
0	Crankshaft and camshaft signals are not synchronized.
1	Evaluation of the crankshaft and camshaft signals has started. ▪ Speed measurement activated (for details on the start time, refer to Basics on Speed Measurement (Crank Parameters) on page 215)
2	Crankshaft angle measurement is being adjusted (the angle counter TCR2 is synchronized with the angle counter of the ACU). ▪ Speed measurement activated
3	Crankshaft and camshaft signals are synchronized. ▪ Speed measurement activated ▪ Crankshaft angle measurement activated
4	Crankshaft and camshaft signals are synchronized. ▪ Speed measurement activated ▪ Crankshaft angle measurement activated ▪ Generation of injection and ignition pulses activated
5	(Only possible if no RPCU_CAM_TPU_BLx block exists in the model) Crankshaft and camshaft signals are not synchronized and are not intended to be. ▪ Speed measurement activated
15	(Only possible if the following error counters have been incremented: Crank speed error counter, or Cam event error counter - refer to the RPCU_ENG_STATUS_WT_BLx block) ▪ Speed measurement disabled ▪ Crankshaft angle measurement disabled ▪ Generation of injection and ignition pulses disabled This status can be left only if: ▪ The application is restarted. ▪ The crankshaft and camshaft signals are resynchronized. This happens when the trigger inport of the RPCU_CRANK_SETUP_TPU_BLx block receives a trigger signal.

Engine status due to standard engine start

Suppose the application is running, and the engine starts rotating. Crankshaft and camshaft signals are both measured. The following engine statuses are passed through:

Actual Speed	Direction	Status[1)]
$0 \leq$ speed < 5 rpm Measured speed = 0	0 (either) or[2)] 1 (forward)	0
$5 \leq$ speed < 10 rpm Measured speed = 0	1 (forward)	0
$5 \leq$ speed < 10 rpm Measured speed = (actual) speed	1 (forward)	1

Actual Speed	Direction	Status[1]
$5 \leq$ speed < 10 rpm Measured speed = (actual) speed	1 (forward)	2
$5 \leq$ speed < 10 rpm Measured speed = (actual) speed	1 (forward)	3
Speed $\geq$ 10 rpm Measured speed = (actual) speed	1 (forward)	4

[1] The numbers relate to a crankshaft wheel specification that is based on a wavetable.

[2] Direction is set to 1 at every relevant crank signal edge and is reset to 0 after crank signal timeout (due to low speed).

Engine status due to missing RPCU_CAM_TPU_BLx block

Suppose the application is running, and the engine starts rotating. Only the crankshaft signal is measured. The following engine statuses are passed through:

Actual Speed	Direction	Status[1]
$0 \leq$ speed < 5 rpm Measured speed = 0	0 (either) or[2] 1 (forward)	0
Speed $\geq$ 5 rpm Measured speed = (actual) speed	1 (forward)	5

[1] The numbers relate to a crankshaft wheel specification that is based on a wavetable.

[2] Direction is set to 1 at every relevant crank signal edge and is reset to 0 after crank signal timeout (due to low speed).

Engine status due to reverse crankshaft rotation

Suppose the reverse crankshaft operation mode is activated and the engine is in normal operation. Then the engine slows down, rotates backwards, and rotates forwards again. The following engine statuses are passed through:

Actual Speed	Direction	Status[1]
Speed $\geq$ 5 rpm Measured speed = (actual) speed Engine rotates forwards (normal operation).	1 (forward)	4
$0 \leq$ speed < 5 rpm Measured speed = 0	0 (either) or[2] 1 (forward)	3

Actual Speed	**Direction**	**Status**[1]
Engine slows down. **Note** If reverse crankshaft operation mode is disabled, synchronization is lost (engine status = 0).		
Speed = 0 Engine stops.	0 (either)	3
0 > speed ≥ -5 rpm Measured speed = 0 Engine rotates backwards.	0 (either) or[3] 2 (reverse)	3
Speed< -5 rpm Measured speed = 0 Engine rotates backwards.	2 (reverse)	3
0 > speed ≥ -5 rpm Measured speed = 0 Engine rotates backwards.	0 (either) or[3] 2 (reverse)	3
Speed = 0 Engine stops.	0 (either)	3
0 ≤ speed < 5 rpm Measured speed = 0 Engine begins rotating (forwards). **Note** The angle counter continues counting at the position at which reverse rotation started, after one engine cycle at maximum.	0 (either) or[2] 1 (forward)	3
5 ≤ speed < 10 rpm Measured speed = (actual) speed Engine rotates forwards.	1 (forward)	3
Speed ≥ 10 rpm Measured speed = (actual) speed Engine rotates forwards (normal operation).	1 (forward)	4

[1] The numbers relate to a crankshaft wheel specification that is based on a wavetable.

[2] Direction is set to 1 at every relevant crank signal edge and is reset to 0 after crank signal timeout (due to low speed).

[3] Direction is set to 2 at every relevant crank signal edge and is reset to 0 after crank signal timeout (due to low speed).

Note

There is a hysteresis regarding activation and deactivation of ignition/injection signals:
- Speed increases:
  Injection/ignition signals are activated above 10 rpm.
- Speed decreases:
  Injection/ignition signals are deactivated below 5 rpm.

Related topics

Basics

HowTos

References

dsrpcu_crank_read (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
RPCU_CRANK_SETUP_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
RPCU_ENG_SPEED_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
RPCU_ENG_STATUS_WT_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Specifying Speed, Position, and Status Measurement

Where to go from here

Information in this section

How to Measure the Engine Speed

Objective

If you want to monitor the speed of an engine as crankshaft revolutions per minute (rpm), you have to perform the following steps.

Preconditions

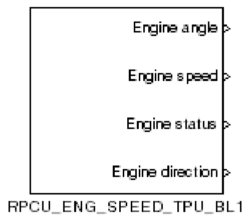
Before you measure engine speed, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.

Method

To measure the engine speed

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_ENG_SPEED_TPU_BLx block to the model.



2 Open the block, and select the Unit page.

3 Configure the settings of the Unit page to match their counterpart settings on the Unit page of the RPCU_CRANK_SETUP_TPU_BLx block.

Result

The measured engine speed can be read at the Speed outport of the RPCU_ENG_SPEED_TPU_BLx block when the RTI model is running.

Note

The output signal must be connected to a Gain block before it can be used by ControlDesk 3.x or ControlDesk.

Related topics

Basics

Basics on Speed Measurement (Crank Parameters)........215

HowTos

How to Measure the Engine Position........229

How to Measure the Engine Position

Objective

If you want to monitor the engine's crankshaft position via angle measurement, you have to perform the following steps.

Definition of the engine position

The engine position is represented by an angle value which is derived from the time counter register TCR2. For detailed information. Refer to Processing the Crankshaft Signal on page 147.

Preconditions

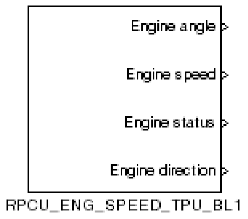
Before you measure engine position, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.

Method

To measure the engine position

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_ENG_SPEED_TPU_BLx block to the model.



2 Open the block, and select the Unit page.

3 Configure the settings of the Unit page to match their counterpart settings on the Unit page of the RPCU_CRANK_SETUP_TPU_BLx block.

Result

The measured engine position can be read at the Angle outport of the RPCU_ENG_SPEED_TPU_BLx block when the RTI model is running.

Note

The output signal must be connected to a Gain block before it can be used by ControlDesk 3.x or ControlDesk.

Related topics

HowTos

How to Measure the Engine Status

Objective

To monitor the engine behavior.

Engine status

The engine status is indicated by numbers whose meanings depend on the crankshaft wheel specification (wavetable or parameters), refer to Basics on the Engine Status (Crank Parameters) on page 218 and Basics on the Engine Status (Crank Wavetable) on page 223.

Preconditions

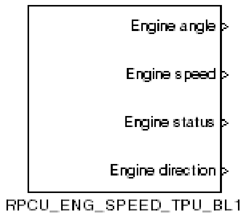
Before you measure engine status, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.

Method

To measure the engine status

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_ENG_SPEED_TPU_BLx block to the model.



2 Open the block, and select the Unit page.

3 Configure the settings of the Unit page to match their counterpart settings on the Unit page of the RPCU_CRANK_SETUP_TPU_BLx block.

Result

The engine status can be read at the status outport of the RPCU_ENG_SPEED_TPU_BLx block when the RTI model is running.

Note

The output signal must be connected to a Gain block before it can be used by ControlDesk 3.x or ControlDesk.

Related topics

HowTos

How to Analyze Synchronization Errors

Objective

To find out why synchronization is lost or cannot be achieved.

Error counters

If synchronization is lost or cannot be achieved there are error counters that give an indication of the reason.

Preconditions

Before you measure engine status, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.

- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.

Possible methods

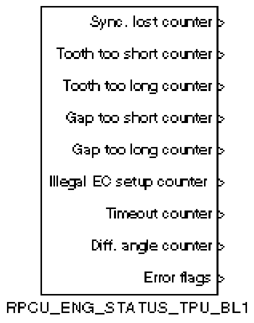
There are two methods depending on how the crankshaft wheel is specified:

- If the crankshaft wheel specification is based on parameters, refer to Method 1.
- If the crankshaft wheel specification is based on a wavetable, refer to Method 2.

Method 1

To monitor the synchronization status (using parameters)

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_ENG_STATUS_TPU_BLx block to the model.



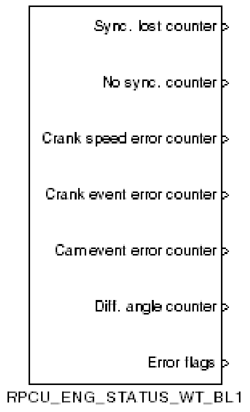
2 Open the block and select the Unit page.

3 Configure the settings of the Unit page to match their counterpart settings on the Unit page of the RPCU_CRANK_SETUP_TPU_BLx block.

Method 2

To monitor the synchronization status (using a wavetable)

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_ENG_STATUS_WT_BLx block to the model.



2 Open the block and select the **Unit** page.

3 Configure the settings of the **Unit** page to match their counterpart settings on the **Unit** page of the RPCU_CRANK_SETUP_TPU_BLx block.

Result

The synchronization status can be read at the status outport of the RPCU_ENG_SPEED_TPU_BLx block when the RTI model is running.

Related topics

Basics

Basics on the Synchronization of the Crankshaft and Camshaft Signals (Wavetable-Based)........ 150

HowTos

How to Measure the Engine Position........ 229
How to Measure the Engine Speed........ 228
How to Measure the Engine Status........ 230

Examples

Examples of Worst-Case Synchronization Times........ 155

References

RPCU_ENG_STATUS_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_ENG_STATUS_WT_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Measuring Camshaft Phase Shifts

Where to go from here

Information in this section

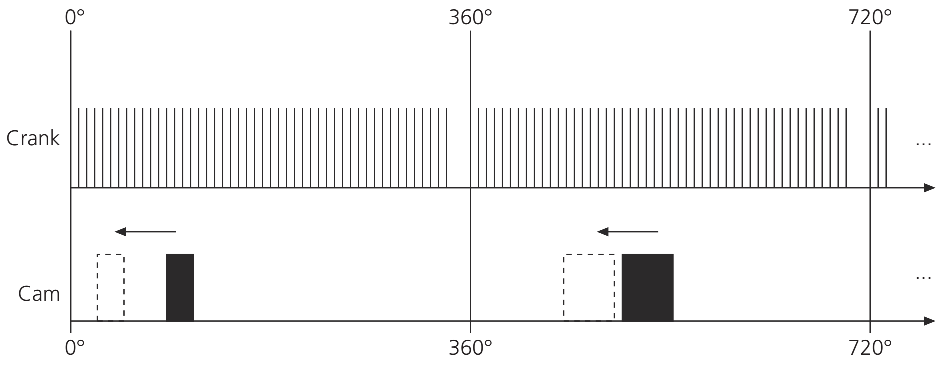
Basics on Camshaft Phase Shift Measurement

Objective

Provides information on the configuration options for the camshaft phase shift measurement.

Definition of a camshaft phase shift

Camshaft phase shift means an angular displacement of a camshaft relative to the coupled crankshaft, triggered by an external mechanism. The entire pulse pattern for the camshaft signal is shifted. A camshaft phase shift is commonly a function of speed and load. The graphic below shows a negative phase shift, i.e., the signal edges occur earlier than for the initial camshaft position.

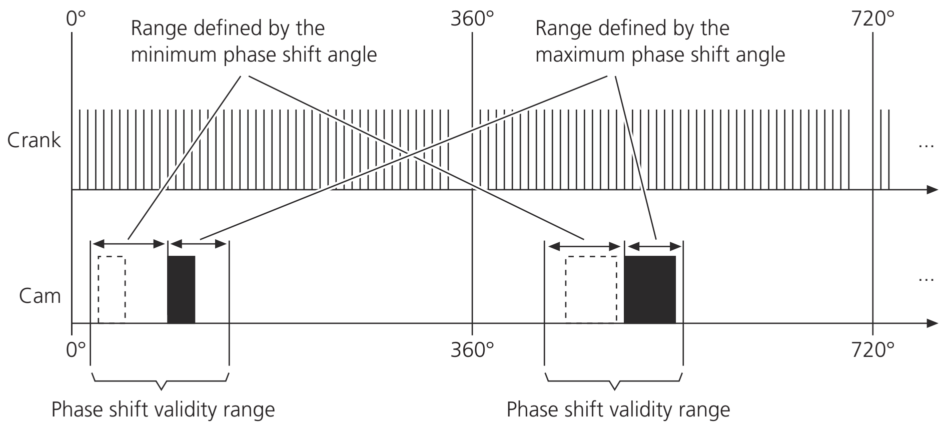


Configuring the measurement

Phase shift validity range Erroneous edges in the camshaft signal, for example, due to noise, corrupt the measurement of the camshaft phase shift. To filter out such invalid measurement values, you must define a phase shift validity range by specifying the following two parameters:

[Minimum phase shift angle ... Maximum phase shift angle].

For the illustration below, it is assumed that the **Edge polarity** is set to **Rising edge**. Thus, the two parameters relate to this signal edge.


0°
Range defined by the minimum phase shift angle
360°
Range defined by the maximum phase shift angle
720°
Crank
...
Cam
...
0°
360°
720°
Phase shift validity range
Phase shift validity range

Computation of the phase shift

Based on the pulse pattern you specified for the camshaft markers (refer to How to Set Up Camshaft Signal Processing on page 207), the system "knows" the angle values of the signal edges that are to come.

Signal check A check is performed for each camshaft signal edge that is detected: Does the angle value of the detected signal edge fit the validity range of the signal edge N that is to come next?

- Yes:

  The phase shift is computed. The **RPCU_CAM_TPU_BLx** block outputs the following signals:

Block Outport	Signal Value
Phase	New phase shift result
Error	Last counter value
Status	1 (= new value)

- No:

  The specified camshaft markers are subsequently tried in descending order (N-1, N-2, ...) until one matches the measured signal edge.

  The phase shift is computed. The **RPCU_CAM_TPU_BLx** block outputs the following signals:

Block Outport	Signal Value
Phase	New phase shift result
Error	+1
Status	1 (= new value)

If no match is found, the last measured phase shift is returned. The RPCU_CAM_TPU_BLx block outputs the following signals:

Block Outport	Signal Value
Phase	Last phase shift result
Error	+1
Status	0 (= old value)

- If too many measurements are faulty, the phase shift measurement is suspended. The RPCU_CAM_TPU_BLx block outputs the following signals:

Block Outport	Signal Value
Phase	0°
Error	Last counter value
Status	2 (= measurement suspended)

  The next signal edge that unambiguously fits the validity range of any camshaft marker is used to resynchronize the camshaft phase shift measurement.

Related topics

Examples

How to Set Up Camshaft Phase Shift Measurement

Objective

To set up the measurement of camshaft phase shift.

Measuring a camshaft phase shift

The camshaft phase shift can be read at the Phase outport. The camshaft phase shift of a camshaft wheel can be measured even if a camshaft wheel is excluded from synchronization (refer to How to Exclude a Camshaft Wheel from Synchronization on page 213).

Note

After the camshaft phase shift is measured, the slave sends the result to the master. However, no interrupt is generated.

Preconditions

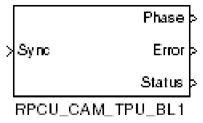
Before you start setting up camshaft signal processing, the following preconditions must be met:

- The controller model is opened.
- Camshaft signal processing is set up. Refer to How to Set Up Camshaft Signal Processing on page 207.

Method

To set up camshaft phase shift measurement

1 Open the related RPCU_CAM_TPU_BLx block.



2 Select the Cam Phase Measurement page.

3 Select the Enable phase measurement checkbox.

4 Specify the camshaft validity range by specifying the Minimum phase shift angle and the Maximum phase shift angle.

Result

Phase shift measurement is implemented for the camshaft wheel that is specified by the RPCU_CAM_TPU_BLx block.

Related topics

Basics

Basics on Camshaft Phase Shift Measurement...234

Examples

Example of Detecting Invalid Camshaft Signals...237

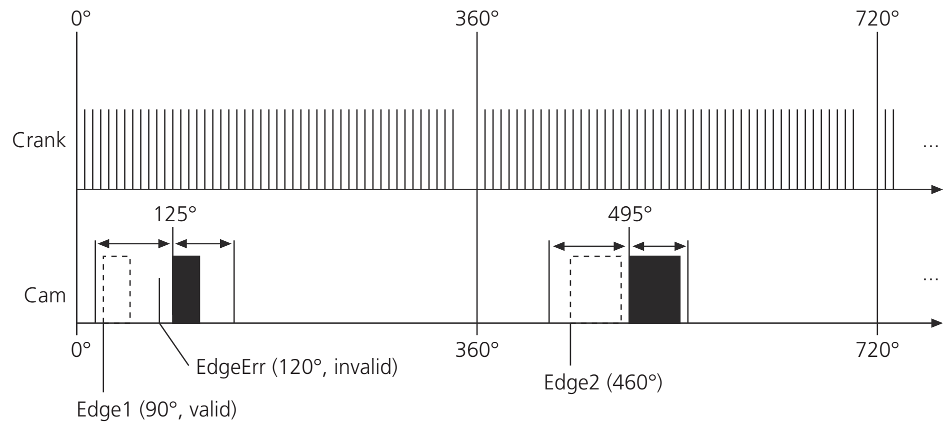
Example of Detecting Invalid Camshaft Signals

Objective

Provides information on the camshaft phase shift validity range.

Scenario

Suppose the camshaft phase shift equals -35° and an invalid signal edge occurs within the validity range as illustrated below.



Computation of the phase shift

The camshaft phase shift is computed as follows:

1. **Edge1** is measured. Does **Edge1** fit the validity range of the signal edge that is to come next (**Edge1**)?
   - Yes.

     The phase shift is computed to -35°.
2. **EdgeErr**, the invalid signal edge, is measured. Does **EdgeErr** fit the validity range of the signal edge that is to come next (**Edge2**)?
   - No.
   - Does **EdgeErr** fit the validity range of **Edge1** ?
   - Yes.

     The phase shift is computed to -5°.
3. **Edge2** is measured. Does **Edge2** fit the validity range of the signal edge that is to come next (**Edge2**)?
   - Yes.

     The phase shift is computed to -35°.

The measurement quickly recovers after the invalid signal edge occurs. Only one measurement value is wrong.

Related topics

Basics

Generating Injection and Ignition Pulses

Objective

The RapidPro Control Unit provides special output features that you can use for generating injection and ignition pulses.

Where to go from here

Information in this section

Information in other sections

Specification of Injection and Ignition Pulse Patterns

Objective

This section provides basic information on generating injection and ignition pulses.

Synchronization

If injection and ignition pulse generation is enabled, the RapidPro Control Unit starts generating injection and ignition pulses as soon as angle measurement is synchronized.

Pulse specification

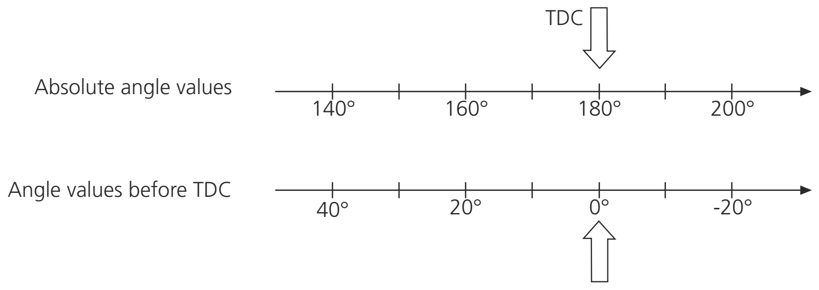
Injection and ignition pulses are specified for each cylinder separately by the following parameters:

Pulse	Parameters
Injection	▪ Start angle ▪ Fuelling time
Ignition	▪ Start angle ▪ End angle

It is possible to specify a sequence of injection and ignition pulses for a cylinder (vectorized input of the pulse parameters).

TDC as reference

All angle values that are used to specify injection and ignition pulses are relative to "before TDC". You get these values by subtracting the current absolute angle value from the absolute TDC angle value.



Example Suppose ignition must start at 140°, and TDC be 180°. You have to specify +40° as the start angle (140° = 180° - (+40°)).

Updating of the pulse specification at run time

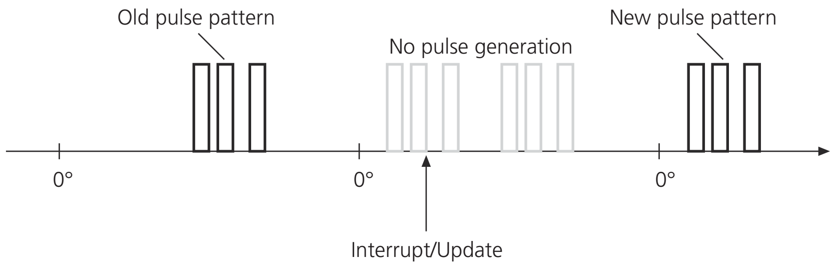
Injection and ignition pulse patterns are initially specified in the block dialog. These initial values are overwritten by specific signals connected to the inports of the RPCU_INJ_IGN_TPU_BLx during run time.

The parameters of both injection and ignition pulse patterns make up one single parameter set. Thus, the specifications of injection and ignition pulse patterns can only be updated as a whole. The pulse patterns can be updated as long as the next pulse generation has not started. A change of signal values during pulse generation does not affect the pulses currently being generated. However, you must consider the time required for master-to-slave communication.

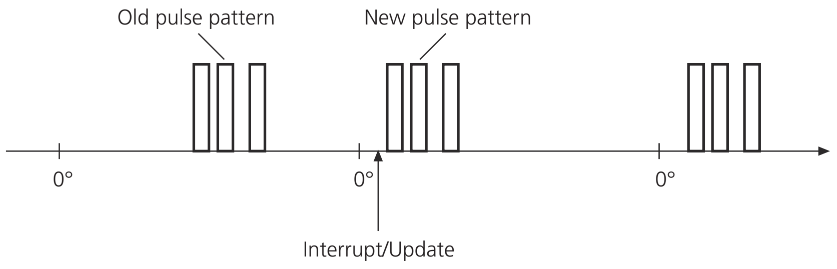
Angle-triggered subsystem If the RPCU_INJ_IGN_TPU_BLx resides in a subsystem triggered by an angle interrupt (e.g., refer to DemoRPCUEngineControl Model on page 381) and you want to update, for example, the injection pulse pattern at run time, the angle value of the interrupt must be smaller than the smallest angle value of the new injection pulse pattern (in addition, consider the time needed for master-to-slave communication).

Otherwise, pulse generation fails completely during one engine cycle and the update takes effect in the next but one engine cycle.

Here you see an illustration of an *inappropriate* angle interrupt which is not consistent with the new pulse pattern:



Here you see an illustration of an *appropriate* angle interrupt which is consistent with the new pulse pattern:



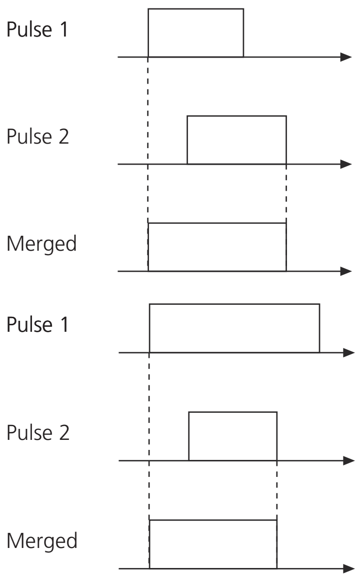
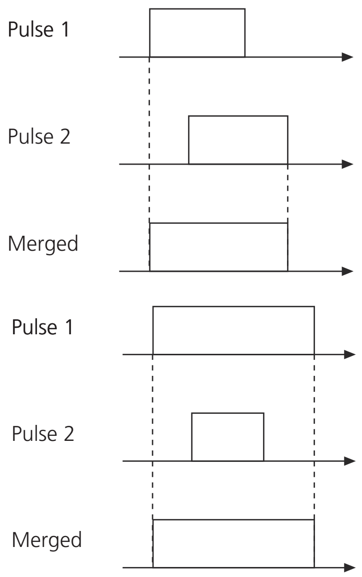
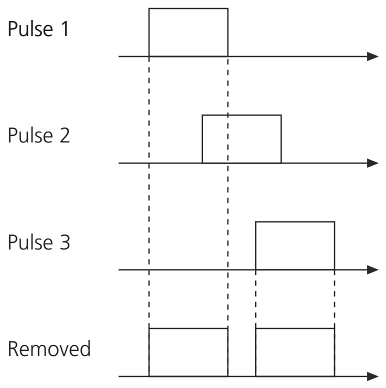
Pulse generation

Injection and ignition pulses can be generated as high or low active pulses. Generation of the pulses always starts with the first pulse specified, even when synchronization got lost and is reestablished. If the crankshaft accelerates or decelerates, it takes one period to adapt the pulse generation.

Overlapping injection and ignition pulses

For two or more overlapping injection and ignition pulses, you can select between the following two kinds of pulse handling: “Merge” and “Remove” (refer to Overlap mode):

Pulse Handling	Injection Pulses	Ignition Pulses
“Merge”	One new pulse is created: ▪ Start angle = Start angle of the pulse that started at first	One new pulse is created: ▪ Start angle = Start angle of the pulse that started at first ▪ End angle = Largest end angle

Pulse Handling	Injection Pulses	Ignition Pulses
	▪ End angle = End angle of the pulse that started at last	
	 Pulse 1 Pulse 2 Merged Pulse 1 Pulse 2 Merged	 Pulse 1 Pulse 2 Merged Pulse 1 Pulse 2 Merged
"Remove"	From left to right (ascending angle values), if a pulse overlaps the preceding pulse it is removed	
		

Note

The following limitations apply:

- Injection pulses and Remove mode:
  The following values apply for the lower limit of the temporal distance between two adjacent injection pulses Δt_{min}:

TPU timer resolution < 2 µs	Δt_{min} = 2 µs
TPU timer resolution ≥ 2 µs	Δt_{min} = 2 µs + TPU timer resolution

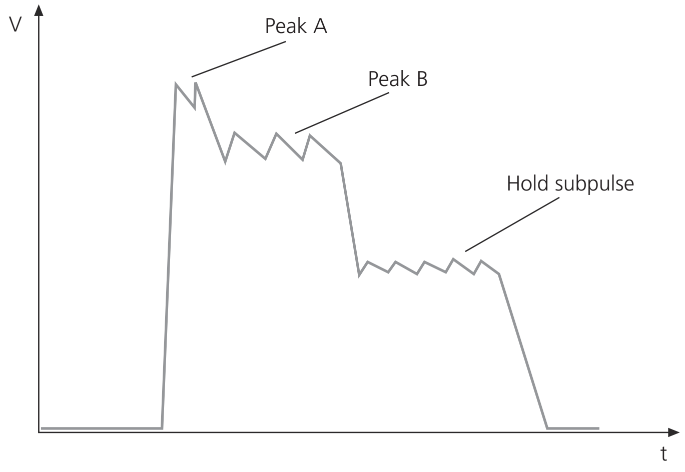
- Ignition pulses:
  If two pulses have a too short distance in between them (depends on engine speed and MPC565 load), ignition pulse generation becomes erroneous. However, if two pulses overlap (distance ≤ 0), there is not any problem.

Refer to Injection and Ignition Reference Data on page 276.

Parts of an injection pulse

An injection pulse can be split into three subpulses:

1. Peak A (Range: 0.0000032 … 0.002 s)
2. Peak B (Range: 0 … 0.002 s)
   0 s means that a B peak does not exist.
3. Hold subpulse (the remainder of the injection pulse)



The peak A is used to generate a high-voltage signal that activates the injection nozzle. The peak B is used to generate a middle-voltage signal that quickly opens the nozzle. The hold subpulse is used to generate a low-voltage signal that keeps the nozzle open.

Note

You can change the duration of the A and B peaks as well as the duration of the complete injection pulse at run time.

You can manipulate the output of the injection subpulses via the two module output channels as follows:

GATE Mode	First Channel	Second Channel
ON	Peak A + Peak B + Hold	[Peak A + Peak B + Hold]$_{\text{inverted}}$
OFF	Peak A + Peak B	Peak B + Hold

Refer to the Enable gate mode parameter.

Error handling

If an error occurs (for example, synchronization lost), injection and ignition pulse generation is stopped immediately.

Related topics

Basics

References

How to Specify Injection Pulses for a Cylinder (Initial Values)

Objective

If you want to generate injection pulses for a specific cylinder, you have to perform the following steps.

One RTI block per cylinder

Each cylinder requires its own RPCU_INJ_IGN_TPU_BLx block, in other words, one RPCU_INJ_IGN_TPU_BLx block specifies the injection/ignition pulse generation of one cylinder.

Preconditions

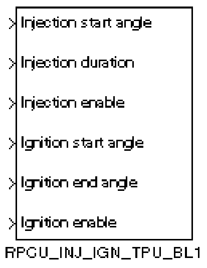
Before you specify injection pulses, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.
- You must know the TDC values of each cylinder. Refer to Definition of top dead center on page 164.

Method

To specify injection pulses for a cylinder (initial values)

1 From the RapidPro **Library: rtirpculib/ENGINE CONTROL**, copy an RPCU_INJ_IGN_TPU_BLx block to the model.



2 Open the block, and select the **Unit** page.

3 From the **Cylinder number** list, choose a number. No other cylinder must be addressed by the same number.

4 In the **Top dead center angle** field, specify the TDC angle.

5 Select the **Injection** page.

6 Select the **Injection enable** checkbox.

7 In the **Output driver selection** list, choose a TPU channel for injection signal output. You have to specify the first, even one. The second, odd one is chosen automatically.

8 Specify the injection pulse parameters.

9 If you want to specify other pulses for the termination case, select the **Termination state** checkbox, and specify the injection pulse parameters.

Result

The injection pulse generation of the addressed cylinder is specified.

Tip

You can update the specification of the pulse pattern at run time, How to Update an Injection/Ignition Pulse Pattern at Run Time on page 249.

Next step

To actually start the pulse generation, you have to implement an RPCU_INJ_IGN_ENABLE_TPU block. Refer to How to Start and Stop Pulse Generation Globally at Run Time on page 247.

Related topics

Basics

HowTos

How to Specify Ignition Pulses for a Cylinder (Initial Values)

Objective

If you want to generate ignition pulses for a specific cylinder, you have to perform the following steps.

One RTI block per cylinder

Each cylinder requires its own RPCU_INJ_IGN_TPU_BLx block, in other words, one RPCU_INJ_IGN_TPU_BLx block specifies the injection/ignition pulse generation of one cylinder.

Preconditions

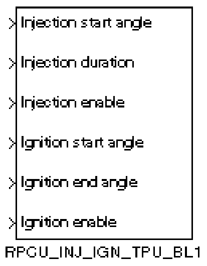
Before you specify ignition pulses, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.
- You must know the TDC values of each cylinder. Refer to Definition of top dead center on page 164.

Method

To specify ignition pulses for a cylinder

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_INJ_IGN_TPU_BLx block to the model.



2. Open the block, and select the **Unit** page.
3. From the **Cylinder number** list, choose a number. No other cylinder must be addressed by the same number.
4. In the **Top dead center angle** field, specify the TDC angle.
5. Select the **Ignition** page.
6. Select the **Ignition enable** checkbox.
7. In the **Output driver selection** list, choose a TPU channel for ignition signal output.
8. Specify the ignition pulse parameters.
9. If you want to specify other pulses for the termination case, select the **Termination state** checkbox, and specify the ignition pulse parameters.

Result

The ignition pulse generation of the addressed cylinder is specified.

Tip

You can update the specification of the pulse pattern at run time, How to Update an Injection/Ignition Pulse Pattern at Run Time on page 249.

Next step

To actually start the pulse generation, you have to implement an RPCU_INJ_IGN_ENABLE_TPU block. Refer to How to Start and Stop Pulse Generation Globally at Run Time on page 247.

Related topics

Basics

Specification of Injection and Ignition Pulse Patterns............ 239

HowTos

How to Specify Injection Pulses for a Cylinder (Initial Values)............ 244
How to Update an Injection/Ignition Pulse Pattern at Run Time............ 249

How to Start and Stop Pulse Generation Globally at Run Time

Objective

Starting pulse generation for all cylinders at once at run time is a mandatory precondition for using the injection/ignition feature.

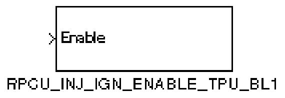
Applied mechanism

All injection and ignition pulse generation (DS1007: related to a specific ECU channel) can be stopped and started at once, that is, all output signals are set to zero.

Method

To start and stop pulse generation globally at run time

1 From the RapidPro **Library: rtirpculib/ENGINE CONTROL**, copy an RPCU_INJ_IGN_ENABLE_TPU_BLx block to the model.



2 Open the block.

The topology ID of the RapidPro hardware used is displayed in the **TopologyID field**.

3 In the **Board/Module number** list, choose the same board number as in the RPCU_INJ_IGN_TPU_BLx block(s).

4 In the **ECU channel** list, choose the same ECU channel number as in the RPCU_INJ_IGN_TPU_BLx block(s).

(Only if you work with a DS1007 modular system)

5 Connect a Boolean signal line to the inport.

Result

A false input signal stops pulse generation globally at run time, and true starts it.

Related topics

HowTos

How to Start and Stop Injection/Ignition Pulse Generation of a Cylinder at Run Time

Objective

If you want to start and stop injection/ignition pulse generation of a cylinder at run time, you have to perform the following steps.

Applied mechanism

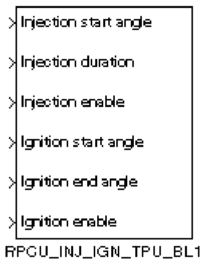
Injection and ignition pulse generation can be disabled for each cylinder separately, that is, the output signal is set to zero.

Method

To start and stop injection/ignition pulse generation of a cylinder at run time

1 Open the RTI model.

2 Connect a Boolean signal line to the Injection/Ignition enable inport of the RPCU_INJ_IGN_TPU_BLx block.



Result

A true input signal starts pulse generation at run time, and false stops it.

Related topics

HowTos

How to Update an Injection/Ignition Pulse Pattern at Run Time

Objective

If you want to update a specific pulse of a pulse pattern at run time, you have to perform the following steps.

Applied mechanism

The specification of an injection and ignition pulse pattern can be updated via the inports of the RPCU_INJ_IGN_TPU_BLx block at run time.

Note

As the specification of injection and ignition pulses allows vectorized input, you can modify as many pulses of one pulse pattern as you want at run time.

Clearance of specific pulses

To clear a specific pulse, its specification must read as follows:

- Injection: Injection duration is zero
- Ignition: Ignition start angle equals Ignition end angle

Preconditions

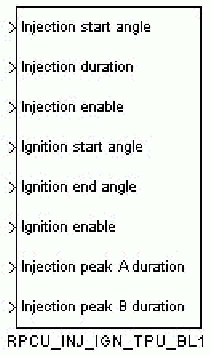
If you want to update the injection peak A and peak B durations at run time, you must have enabled the associated input ports of the RPCU_INJ_IGN_TPU_BLx. Refer to Advanced Page (RPCU_INJ_IGN_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Method

To update an injection/ignition pulse pattern at run time

1 Open the RTI model.

2 Connect a signal line to the desired inport(s) of the RPCU_INJ_IGN_TPU_BLx block.



The incoming signal must have the right dimension (number of pulses).

Result

You can update the injection/ignition pulse pattern at run time.

Related topics

HowTos

How to Disable Injection/Ignition Pulse Generation of a Cylinder

Objective

If you want to disable injection/ignition pulse generation for a specific cylinder, you have to perform the following steps.

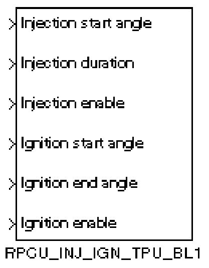
Preconditions

There must be an RPCU_INJ_IGN_TPU_BLx block representing the desired cylinder in the RTI model.

Method

To disable injection/ignition pulse generation

1 Open the RPCU_INJ_IGN_TPU_BLx block.



2 Select the Injection/Ignition page.

3 Clear the Injection/Ignition enable checkbox.

Result

Injection/ignition pulse generation for the addressed cylinder is disabled.

Related topics

HowTos

Generating Angle-Angle-Based Pulses

Objective

The RapidPro Control Unit (TPU) provides special output features that you can use for generating angle-angle-based pulses specified by start and end angles.

Note

Angle-angle-based pulses can be used in a similar way to ignition pulses. You can generate them in parallel with but independently of ignition pulses.

Where to go from here

Information in this section

Information in other sections

Angle-Angle-Based Pulse Generation (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Angle-Angle-Based Pulse Generation (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

Basics on the Generation of Angle-Angle-Based Pulses

Objective

This section provides basic information on generating angle-angle-based pulses.

Pulse pattern

The following parameters are used to specify an angle-angle-based pulse pattern:

- Start angle (range: -360° ... 359.9°; with respect to "before TDC")
- End angle (range: -360° ... 359.9°; with respect to "before TDC")

One pulse pattern can comprise a sequence of angle-angle-based pulses (vectorized input of the angle values). You can change these parameters at runtime.

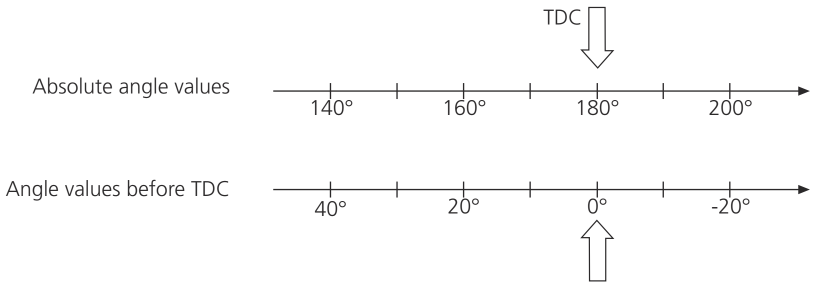
For details on the specification of these parameters, refer to Parameters Page (RPCU_AABP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

Note

Angle-angle-based pulses can be compared favorably with ignition pulses.

TDC as reference

All angle values that are used to specify angle-angle-based pulses are relative to “before TDC”. You get these values by subtracting the current absolute angle value from the absolute TDC angle value.



Example Suppose a pulse must start at 140° and end at 150°, and TDC be 180°. You have to specify +40° as the start angle and +30° as the end angle.

Updating of the pulse specification at run time

Angle-angle-based pulse patterns are initially specified in the block dialog. These initial values are overwritten by specific signals connected to the **Start angle** and **End angle** inports at runtime.

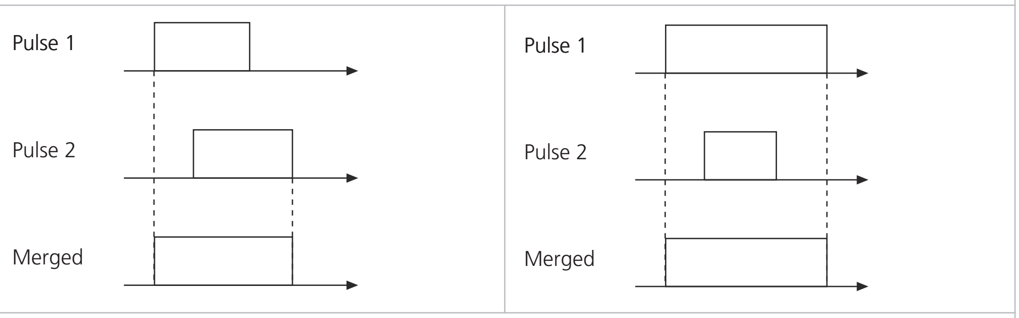
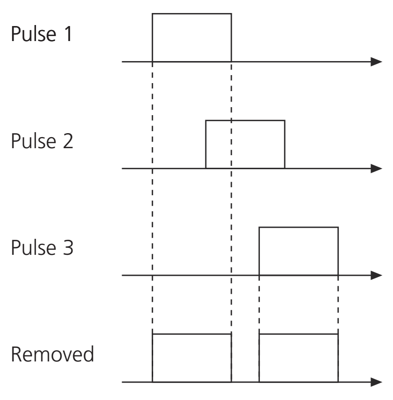
The pulse patterns can be updated as long as the next pulse generation has not started. A change of signal values during pulse generation does not affect the pulses currently being generated. However, you must consider the time for the master-to-slave communication.

Pulse generation

Angle-angle-based pulses are always generated as high active pulses. Generation of the pulses always starts with the first pulse specified, even when synchronization got lost and is reestablished. If the crankshaft accelerates or decelerates, it takes one period to adapt the pulse generation.

Overlapping angle-angle-based pulses

For two or more overlapping angle-angle-based pulses, you can select between the following two kinds of pulse handling: "Merge" and "Remove":

Pulse Handling	Angle-Angle-Based Pulses
"Merge"	One new pulse is created: ▪ Start angle = Start angle of the pulse that started at first ▪ End angle = Largest end angle  Pulse 1 Pulse 2 Merged Pulse 1 Pulse 2 Merged
"Remove"	From left to right (ascending angle values), if a pulse overlaps the preceding pulse it is removed 

Note

The following limitations apply:

- If two pulses have a too short distance in between them (depends on engine speed and MPC565 load), pulse generation becomes erroneous. However, if two pulses overlap (distance ≤ 0), there is not any problem.

Related topics

Basics

Synchronization Details on Parameter-Based Crankshaft Wheel Specifications........................ 158

HowTos

How to Specify Angle-Angle Based Pulses (Initial Values).. 255

How to Specify Angle-Angle Based Pulses (Initial Values)

Objective

If you want to generate angle-angle based pulses, you have to perform the following steps.

One RTI block per pulse pattern

Each pulse pattern requires its own RPCU_AABP_TPU_BLx block.

Preconditions

Before you specify angle-angle based pulses, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.
- You must know the TDC value. Refer to Definition of top dead center on page 164.

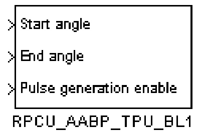
Restriction

Up to 12 RPCU_AABP_TPU_BLx blocks and up to 4 RPCU_SCKNOCK41_SETUP Block blocks are allowed in one model, but they must not exceed an overall total of 12.

Method

To specify an angle-angle based pulse pattern

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_AABP_TPU_BLx block to the model.



2 Open the block, and select the Unit page.

3 Specify the RapidPro unit, and in the Output port selection list, choose a TPU channel for signal output.

4 Select the Parameters page.

5 Configure the angle-angle based pulse generation.

6 If you want to specify a different pulse pattern for the termination case, select the Termination state checkbox, and specify the pulse parameters.

Result

The angle-angle based pulse generation is specified.

Next step

To actually start the pulse generation, you have to implement an RPCU_AABP_ENABLE_TPU_BLx block. Refer to How to Start and Stop Pulse Generation Globally at Run Time on page 256.

Related topics

Basics

Basics on the Generation of Angle-Angle-Based Pulses... 252
RPCU_AABP_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

HowTos

How to Update an Angle-Angle Based Pulse Pattern at Run Time.. 258

How to Start and Stop Pulse Generation Globally at Run Time

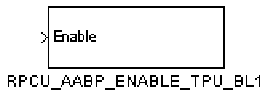
Objective

Starting pulse generation for all channels at once at run time is a mandatory precondition for using the angle-angle-based pulse generation feature.

Method

To start and stop pulse generation globally at run time

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_AABP_ENABLE_TPU_BLx block to the model.



2 Open the block.

The topology ID of the RapidPro hardware used is displayed in the TopologyID field.

3 In the Board/Module number list, choose the same board number as in the RPCU_AABP_TPU_BLx block(s).

4 In the ECU channel list, choose the same ECU channel number as in the RPCU_AABP_TPU_BLx block(s).

(Only if you work with a DS1007 modular system based)

5 Connect a Boolean signal line to the inport.

Result

A false input signal stops pulse generation globally at run time, and true starts it.

Note

The RPCU_AABP_ENABLE_TPU_BLx also affects knock signal measurement. Enabling/disabling angle-angle-based pulse generation also enables/disables knock signal measurement.

Related topics

Basics

RPCU_AABP_ENABLE_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

HowTos

How to Start and Stop Angle-Angle-Based Pulse Generation of a Channel at Run Time..257

How to Start and Stop Angle-Angle-Based Pulse Generation of a Channel at Run Time

Objective

If you want to start and stop angle-angle-based pulse generation of a channel at run time, you have to perform the following steps.

Applied mechanism

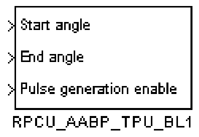
Angle-angle-based pulse generation can be disabled for each channel separately, that is, the output signal is set to zero.

Method

To start and stop angle-angle-based pulse generation of a channel at run time

1 Open the RTI model.

2 Connect a Boolean signal line to the Pulse generation enable inport of the RPCU_AABP_TPU_BLx block that is related to the channel.



Result

A true input signal starts pulse generation at run time, and false stops it.

Related topics

Basics

RPCU_AABP_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

HowTos

How to Start and Stop Pulse Generation Globally at Run Time..256

How to Update an Angle-Angle Based Pulse Pattern at Run Time

Objective

If you want to update a specific pulse of a pulse pattern at run time, you have to perform the following steps.

Applied mechanism

The specification of an angle-angle based pulse pattern can be updated via the inports of the RPCU_AABP_TPU_BLx block at run time.

Note

As the specification of angle-angle based pulses allows vectorized input, you can modify as many pulses of one pulse pattern as you want at run time.

Clearance of specific pulses

To clear a specific pulse of the pulse pattern, its specification must read as follows:

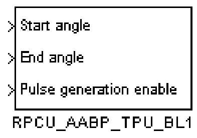
Start angle equals End angle

Method

To update an angle-angle based pulse pattern at run time

1 Open the RTI model.

2 Connect a signal line to the desired inport(s) of the RPCU_AABP_TPU_BLx block.



The incoming signal must have the right dimension (number of pulses).

Result

You can update the angle-angle-based pulse pattern at run time.

Related topics

Basics

Basics on the Generation of Angle-Angle-Based Pulses.. 252
RPCU_AABP_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

HowTos

How to Specify Angle-Angle Based Pulses (Initial Values)... 255

Generating Periodic Angle-Based Trigger Pulses

Objective

To specify an angle-based trigger pulse sequence that is output via the I/O PLD and can be used as an external trigger signal.

Note

In comparison with the generation of pulses via the TPU (e.g. injection pulses), the I/O PLD can generate more pulses per time. However, periodic angle-based trigger pulses cannot be updated at run time.

Where to go from here

Information in this section

Basics on the Generation of Periodic Angle-Based Trigger Pulses

Objective

This section provides basic information on generating periodic angle-based trigger pulses via the I/O PLD.

External trigger source

You can use periodic angle-based trigger pulses as an external trigger source, for example, to invoke A/D conversion. The pulses are output via the I/O PLD. For details on the I/O PLD, refer to General Characteristics of the I/O PLD on page 36.

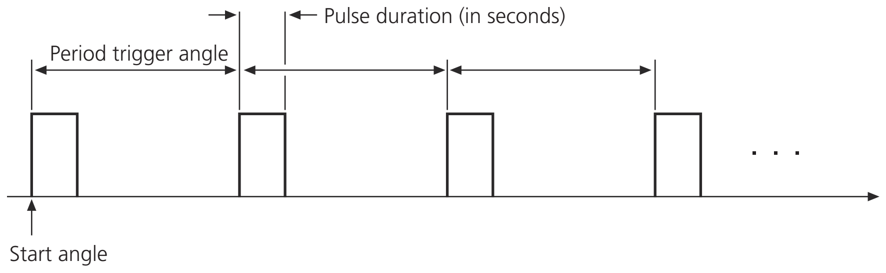
Note

The Bit I/O channel 1 must be routed as BIT_OUT and the channel must not be used by another block of the RTI RPCU blockset.

Pulse pattern

The following parameters are used to specify an periodic angle-based trigger pulse pattern:

- Start angle
- Period trigger angle (distance between two rising edges; range: 0.1° ... 720°)
- Pulse duration (range: 17.8 ns ... 4.55 µs)



Note

You cannot change these parameters during run time.

For details on the specification of these parameters, refer to Advanced Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖).

Restrictions

The following restrictions apply:

- The trigger signal is only generated in a synchronous state, refer to Synchronization Details on Parameter-Based Crankshaft Wheel Specifications on page 158.
- Depending on the period trigger angle, the engine speed must not exceed specific upper limits:
  - 0.1°: 5000 rpm
  - 0.2°: 10000 rpm
  - 0.3°: 15000 rpm
  - 0.4°: 20000 rpm

  Note

  Higher engine speed is basically possible if you decrease pulse duration. However, you must take the specification of the digital output module into account.

- The start angle must be less than the period trigger angle.
- The period trigger angle must divide 720° without any remainder (Period trigger angle * N = 720°).

Related topics

Basics

Synchronization Details on Parameter-Based Crankshaft Wheel Specifications........................ 158

HowTos

How to Implement the Generation of Periodic Angle-Based Trigger Pulses via the I/O PLD.. 262

How to Implement the Generation of Periodic Angle-Based Trigger Pulses via the I/O PLD

Objective

To implement the generation of a periodic angle-based trigger pulse pattern.

Preconditions

Before you implement a periodic angle-based trigger pulse pattern, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.
- The Bit I/O channel 1 is routed as BIT_OUT and the channel is not used by another block.
- You must use I/O PLD firmware version 1.3 or later. To check your I/O PLD firmware version, refer to How to View Hardware Details of Your System (RapidPro – Hardware and Software Getting Started 🕮).

Method

To implement the generation of periodic angle-based trigger pulses via the I/O PLD

1. Open the RPCU_CRANK_SETUP_TPU_BLx block, and select the Advanced page.
2. Select the Enable external trigger signal checkbox.
3. Specify the Period trigger angle. The value must divide 720° without any remainder (Period trigger angle * N = 720°).
4. Specify the Start angle. The start angle must be less than the period trigger angle.
5. Specify the Pulse duration.

Result

The generation of an angle-based periodic trigger pulse pattern is implemented. You can use it as external trigger signal, for example, to invoke A/D conversion.

Related topics

Basics

Generating Angle-Based Interrupts

Objective

The RapidPro Control Unit provides special output features that you can use for implementing angle-based interrupts.

Where to go from here

Information in this section

Information in other sections

Angle-Based Interrupts

Objective

This section provides basic information on generating interrupts that depend on the position of the engine.

Usage as task trigger

You can use angle-based interrupts as task triggers, for example, to update the parameters of injection and ignition pulses:

1. Sensor data is read.
2. New injection and ignition data is calculated.
3. Data is transferred to the slave MPC565.
4. Injection and ignition data is updated at run time.

Note

The RapidPro Control Unit handles angle-based interrupts quickly by hardware.

Occurrence of interrupts

Angle-based interrupts can be generated as follows:

- Interrupts at certain engine positions (position interrupts)

  An interrupt is triggered at a certain engine angle, which can be changed at run time (resolution is 0.1°). 16 position interrupts can be specified at most. A separate RPCU_ANGLE_INT_BLx block is needed for each.
- Interrupts at equidistant angle intervals (periodic interrupts)

  From a certain start angle, an interrupt is triggered at regular angle intervals. Six periodic interrupts can be specified at most. A separate RPCU_ANGLE_INT_BLx block is needed for each.

Note

One position interrupt is represented by one angle value. One periodic interrupt is represented by a sequence of angle values.

Example

Suppose you have specified a start angle of 270° and a period angle of 200°. Interrupts are then generated at the following angle positions: 270°, 470°, 670°, 150°, 350°, … .

Related topics

HowTos

How to Implement a Periodic Interrupt........266
How to Implement a Position Interrupt........264

References

Angle-Based Interrupts Reference Data........280

How to Implement a Position Interrupt

Objective

If you want to generate an interrupt at a certain engine position, you have to perform the following steps.

Specification and update at run time

A position interrupt is initially specified by the Angle parameter, which you can edit in the block dialog. The current angle value of a position interrupt can be overwritten by a signal connected to the Angle position inport, at run time.

Preconditions

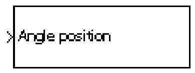
Before you implement a position interrupt, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.

Method

To implement a position interrupt

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy a RPCU_ANGLE_INT_BLx block to the model.



RPCU_ANG_INT_BL1

2 Open the block, and select the Unit page.

3 Configure the settings of the Unit page to match their counterpart settings on the Unit page of the RPCU_CRANK_SETUP_TPU_BLx block.

4 Select the Parameters page.

5 From the Angle number list, choose an ID number for the position interrupt: "1" ... "16".

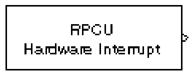
When you have chosen a position interrupt, the block gets an Angle position inport.

6 In the Angle field, specify the initial value of the position interrupt.

7 If you want to update the position interrupt during run time, connect a signal line to the Angle position inport.

The position interrupt is specified.

8 From the RapidPro Library: rtirpculib, copy an RPCU_INTERRUPT_BLx block to the model.



RPCU_INTERRUPT_BL1

9 Open the block.

10 Configure the settings of the Unit page to match their counterpart settings on the Unit page of the RPCU_SETUP_BLx block.

11 In the Interrupt selection field, select the corresponding interrupt specification.

Result

The position interrupt is implemented. You can use the outport of the RPCU_INTERRUPT_BLx block to trigger a subsystem.

Related topics

Basics

HowTos

References

How to Implement a Periodic Interrupt

Objective

If you want to generate interrupts at angle-equidistant intervals, you have to perform the following steps.

Specification

A periodic interrupt is specified by the **Start angle** and **Period angle** parameters, which you can edit in the block dialog. Updating at run time is not possible.

Preconditions

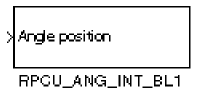
Before you implement a position interrupt, the following preconditions must be met:

- The RapidPro Library: rtirpculib/ENGINE CONTROL is opened.
- The controller model is opened.
- Crankshaft signal processing has been set up. Refer to How to Set Up Crankshaft Signal Processing on page 205.
- Camshaft signal processing has been set up. Refer to How to Set Up Camshaft Signal Processing on page 207.

Method

To implement a periodic interrupt

1 From the RapidPro Library: rtirpculib/ENGINE CONTROL, copy an RPCU_ANGLE_INT_BLx block to the model.



2 Open the block, and select the **Unit** page.

3 Configure the settings of the **Unit** page to match their counterpart settings on the **Unit** page of the RPCU_CRANK_SETUP_TPU_BLx block.

4 Select the **Parameters** page.

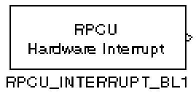
5 From the **Angle number** list, choose an ID number for the periodic interrupt: “Periodic 1” … “Periodic 6”.

   You have chosen a periodic interrupt.

6 In the **Start angle** field, specify the start angle value.

7 In the **Periodic angle** field, specify the interrupt interval, that is, the angular distance between two adjacent interrupts.

8 From the RapidPro **Library: rtirpculib**, copy an RPCU_INTERRUPT_BLx block to the model.



9 Open the block.

10 Configure the settings of the **Unit** page to match their counterpart settings on the **Unit** page of the RPCU_SETUP_BLx block.

11 In the **Interrupt selection** field, select the corresponding interrupt specification.

Result

The periodic interrupt is implemented. You can use the outport of the RPCU_INTERRUPT_BLx block to trigger a subsystem.

Related topics

Basics

Angle-Based Interrupts........263

HowTos

How to Implement a Position Interrupt........264

References

Angle-Based Interrupts Reference Data........280

Error Handling

Error Counters and Interrupt Handling

Error counters

The RapidPro Control Unit provides an error counter that holds statistics about the occurrences of different error types.

For detailed information on the error counters, refer to the RPCU_ENG_STATUS_TPU_BLx block in the RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮.

Interrupt handling

As soon as the ACU detects an error in the synchronized state, synchronization is lost (camshaft phase shift measurement and injection and ignition pulse generation are stopped). In addition, an interrupt is issued to the slave MPC565, which triggers the following actions:

- Start resynchronization
- Update the error counters

Engine Control Reference Data

Objective

This section provides a summary of reference data, for example, ranges and limits, concerning the RapidPro engine control features.

Where to go from here

Information in this section

Information in other sections

Engine Control (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Gives information about the RTI blocks concerned with engine control features.

Engine Control (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Gives information about the RTLib functions concerned with engine control features.

Engine Control I/O Mapping

Objective

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

References

Engine Control RTI/RTLib Support

Objective

The RapidPro engine control features are accessible via RTI blocks and RTLib functions. The section tells you where to look up information about the RTI blocks and RTLib functions.

Crankshaft signal

You can access the crankshaft signal measurement feature via the RTI blockset and the RTLib functions as follows:

- Crankshaft Signal Measurement (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- Crankshaft Signal Measurement (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

Camshaft signal

You can access the camshaft signal measurement feature via the RTI blockset and the RTLib functions as follows:

- Camshaft Signal Measurement (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- Camshaft Signal Measurement (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

Engine measurement

You can access the engine measurement feature via the RTI blockset and the RTLib functions. For details, see:

- Engine Speed and Status (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- Engine Status (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

Note

When working with RTLib, engine speed is computed via the dsrpcu_tpu_crank_pm_init function. Refer to Crankshaft Signal Measurement (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮).

Injection and ignition

You can access the injection and ignition feature via the RTI blockset and the RTLib functions. For details, see:

- Injection and Ignition Pulse Generation (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- Injection and Ignition Pulse Generation (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

Angle-based interrupts

You can access the angle-based interrupt feature via the RTI blockset and the RTLib functions. For details, see:

- Interrupt Handling (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- Interrupt Handling (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

Related topics

References

Crankshaft Signal Reference Data

Objective

This section provides a summary of reference data concerning crankshaft signal measurement.

TPU characteristics

The crankshaft signal measurement feature is only supported by the time processor unit (TPU). Its characteristics are:

TPU Characteristics	TPU A, B, C
Channel usage	3 channels used, regardless of TPU unit: (A, B, or C) ▪ Ch01: Reset of angle counter TCR2 (used internally) ▪ Ch02: Period measurement via TCR1 (used internally) ▪ Ch x: Capture of the crankshaft signal (x = 3 … 15)
Period measurement (TCR1)	The resolution and range depend on prescaler settings. Refer to Prescaling on a TPU on page 54

TPU Characteristics	TPU A, B, C
Angle measurement (TCR2)	Resolution: 0.1° Range: 0° … 719.9°

For detailed information on the TPU characteristics. Refer to General Characteristics of a TPU on page 33.

Note

The TPU unit (A, B, or C) which the crankshaft signal is routed to performs the engine speed measurement.

Supported crankshaft wheels

The following kinds of crankshaft wheels are supported (refer to Possible Crankshaft Wheels on page 144):

- Symmetric
- Asymmetric (arbitrary design)

Supported crankshaft signals

The following kinds of crankshaft signals are supported:

- Active signals (both low and high active signals can be processed, it is not necessary to invert them externally)
- Passive signals

Note

An external inversion of the crankshaft signal, for example, by ConfigurationDesk for RapidPro, is even not allowed.

Each signal type requires a specific SC module. For further information on these modules, refer to the *RapidPro System Installation and Configuration* document.

Related topics

Basics

HowTos

References

Camshaft Signal Reference Data

Objective

This section provides a summary of reference data concerning camshaft signal measurement.

TPU characteristics

The camshaft signal measurement feature is only supported by the time processor unit (TPU). Its characteristics are:

TPU Characteristics	TPU A, B, C
Channel usage	2 channels used, regardless of TPU unit: (A, B, or C) ▪ Ch01: Reset of angle counter TCR2 (internally used) ▪ Ch x: Capture of the camshaft signal (x = 2 or 3 … 16) If the RPCU_CRANK_TPU_BLx block uses ▪ the same TPU unit, the TPU channels Ch01 and Ch02 are used internally and thus not available (x = 3 … 16) ▪ another TPU unit, the TPU channel Ch01 is used internally and thus not available (x = 2 … 16)
Angle values in general	Resolution: 0.1° Range: 0° … 719.9°
Synchronization	Required speed: 5 … 10000 Rpm (using a wavetable) 40 … 20000 Rpm (using parameters)
Camshaft phase shift	Possible range: -360° … +359.9°

For detailed information on the TPU characteristics. Refer to General Characteristics of a TPU on page 33.

Supported camshaft signals

The following kinds of camshaft signals are supported:

- Active signals (both low and high active signals can be processed, it is not necessary to invert them externally)
- Passive signals

Note

An external inversion of the camshaft signal, for example, by ConfigurationDesk for RapidPro, is even not allowed.

Each signal type requires a specific SC module. For more information on these modules, refer to the *RapidPro System Installation and Configuration* document.

Related topics

Basics

HowTos

References

Engine Speed Reference Data

Objective

This section provides a summary of reference data concerning engine speed measurement.

TPU characteristics

The speed measurement feature is supported by the time processor unit (TPU). Its characteristics are:

TPU Characteristics	TPU A, B, C	
Period measurement (TCR1)	Resolution depends on the TPU prescaler value of TCR1:	
	TCR1	*Resolution [ns]*
	2	35.7
	4	71.4
	8	143
	14	250
	28	500
	42	750
	56	1000
	84	1500
	112	2000
	168	3000
	224	4000

TPU Characteristics	TPU A, B, C	
	336	6000
	448	8000

For detailed information on the TPU characteristics. Refer to General Characteristics of a TPU on page 33.

Available engine speed

The available engine speed (Rpm: revolutions per minute) depends on the crankshaft wheel specification (refer to Basics on Crankshaft Wheel Specification on page 201) as follows:

Crankshaft Wheel Specification	Rpm_{min} ... Rpm_{max}
Wavetable	0 ... 10000
Parameters	40 ... 20000

Note

For a parameter-specified crankshaft wheel, the available engine speed does not depend on the TPU prescaler setting (TCR1), though the speed resolution does depend on it.

Related topics

Basics

HowTos

References

Injection and Ignition Reference Data

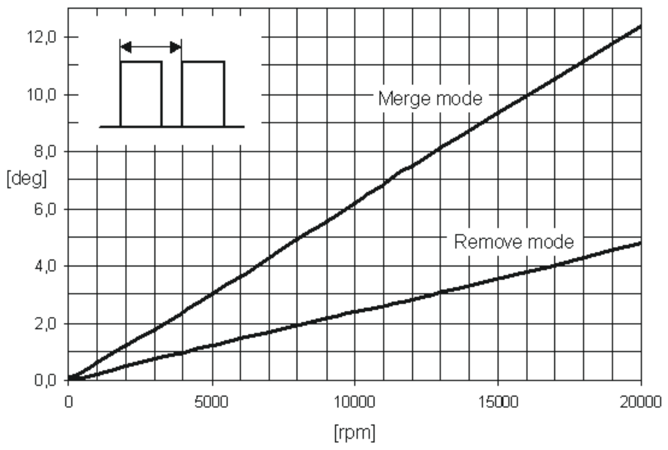
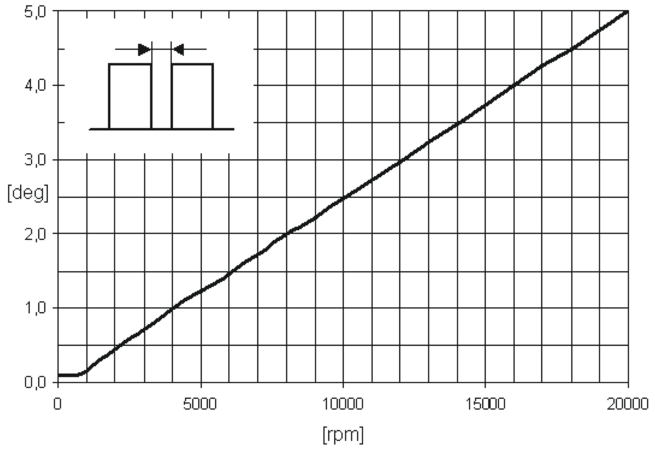
Objective

This section provides a summary of reference data concerning injection and ignition pulse generation.

TPU characteristics

The injection and ignition pulse generation feature is only supported by the time processor unit. Its characteristics are:

Characteristics	TPU A, B, C
Channel usage (injection)	2 channels used, regardless of TPU unit: (A, B, or C) ▪ Ch x: Output of the injection pickup pulse (x = 2, 4, 6, … 14) ▪ Ch x+1: Output of the injection hold pulse (automatically chosen) See also the crankshaft channel usage (TPU characteristics on page 271) and the camshaft channel usage (TPU characteristics on page 273). Crankshaft and camshaft signals must use channels with a smaller channel number.
Channel usage (ignition)	1 channel used, regardless of TPU unit: (A, B, or C) ▪ Ch x: Output of the ignition pulse (x = 2 … 16)
Resolution of angle values in general (TCR2)	Always 0.1° (regardless of any prescaler values) Range: -360.0° ... +359.9° **Note** All angle values are relative to “before TDC”. Suppose ignition must start at 140°, and TDC be 180°. You have to specify +40° as the start angle.
Resolution and range of injection pulses	Resolution depends on the TPU prescaler value of TCR1: *TCR1 : Resolution [ns]* 2: 35.7 4: 71.4 8: 143 14: 250 28: 500 42: 750 56: 1000 84: 1500 112: 2000 168: 3000 224: 4000 336: 6000 448: 8000 The range of the injection pulse length is [0 … 1.0 s], the range of the pickup pulse length is (3.2 µs … 2 ms).

Characteristics	TPU A, B, C
	For the Remove mode, the lower limit of the temporal distance between two adjacent injection pulses Δt_{min} depends on the TPU timer resolution, as follows: ▪ Resolution < 2 µs: Δt_{min} = 2 µs ▪ Resolution ≥ 2 µs: Δt_{min} = 2 µs + resolution
Minimum distance required between consecutive injection pulses	If multiple injection pulses are specified, there must be a minimum distance between them, otherwise pulse generation might lead to unpredictable results. 
Minimum distance required between consecutive ignition pulses	If multiple ignition pulses are specified, there must be a minimum distance between them, otherwise pulse generation might lead to erroneous results. However, if the pulses overlap (distance ≤ 0°), there is not any problem. 
Cylinder range	1 … 12
TDC angle range	0° … 719.99°
Pulse polarity	High (factory setting)

Characteristics	TPU A, B, C
Number of pulses	Up to 15 injection and ignition pulses per channel and engine cycle can be generated (that is, for one cylinder). Start angles must be strictly monotonic increasing.
Range overflow	Pulses can overflow into the next 720° cycle, they are not cropped.
Disabling pulse generation	Pulse generation can be disabled at run time as follows: ▪ All cylinders at once ▪ Specific cylinders ▪ Specific pulses of a pulse pattern (refer to How to Update an Injection/Ignition Pulse Pattern at Run Time on page 249)
Updating the pulse specification	The parameters of both injection and ignition pulses make up one parameter set which can be updated only as a whole.
Overlapping pulses	Two options are available: "Merge" and "Remove". Refer to Overlapping injection and ignition pulses on page 241.

For detailed information on the TPU characteristics. Refer to General Characteristics of a TPU on page 33.

Required modules

An SC module (digital out), or a power stage module which amplifies the ignition and injection pulses, is required. For further information on these modules, refer to the RapidPro System Hardware Reference 📖.

Related topics

Basics

General Characteristics of a TPU...33
Specification of Injection and Ignition Pulse Patterns...239

HowTos

How to Specify Ignition Pulses for a Cylinder (Initial Values)...246
How to Specify Injection Pulses for a Cylinder (Initial Values)...244
How to Update an Injection/Ignition Pulse Pattern at Run Time...249

References

Angle-Based Interrupts Reference Data...280
Camshaft Signal Reference Data...273
Crankshaft Signal Reference Data...271
Engine Control I/O Mapping...269
Engine Control RTI/RTLib Support...270
Engine Speed Reference Data...274

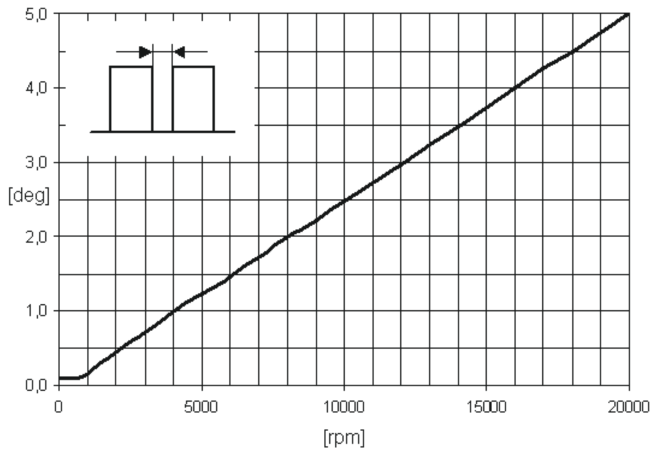
Angle-Angle-Based Pulses Reference Data

Objective

This section provides a summary of reference data concerning angle-angle-based pulse generation.

TPU characteristics

The angle-angle-based pulse generation feature is only supported by the time processor unit. Its characteristics are:

Characteristics	**TPU A, B, C**
Channel usage	1 channel used, regardless of TPU unit: (A, B, or C) ▪ Ch x: Output of the angle-angle-based pulse pattern (x = 2 … 16)
Resolution of angle values in general (TCR2)	Always 0.1° (regardless of any prescaler values) Range: -360.0° ... +359.9° Note: All angle values are relative to "before TDC". Suppose a pulse must start at 140° and end at 150°, and TDC be 180°. You have to specify +40° as the start angle and +30° as the end angle.
Minimum distance required between consecutive pulses	If multiple pulses are specified, there must be a minimum distance between them, otherwise pulse generation might lead to erroneous results. However, if the pulses overlap (distance ≤ 0°), there is not any problem. 
TDC angle range	0° … 719.9°
Pulse polarity	High (factory setting)
Number of pulses	Up to 15 pulses per channel and engine cycle can be generated. Start angles must be strictly monotonic increasing.
Range overflow	Pulses can overflow into the next 720° cycle, they are not cropped.
Disabling pulse generation	Pulse generation can be disabled at run time as follows: ▪ All channels at once ▪ Specific channels ▪ Specific pulses of a pulse pattern (start angle equals end angle)

Characteristics	TPU A, B, C
Overlapping pulses	Two options are available: "Merge" and "Remove". Refer to Overlapping injection and ignition pulses on page 241.

For detailed information on the TPU characteristics. Refer to General Characteristics of a TPU on page 33.

Required modules

An SC module (digital out), or a power stage module which amplifies the angle-angle-based pulses, is required. For further information on these modules, refer to the *RapidPro System Installation and Configuration* document.

Related topics

Angle-Based Interrupts Reference Data

Objective

This section provides a summary of reference data concerning the generation of angle-based interrupts.

Ranges

Position interrupts These are the ranges for position interrupts:

Parameter	Range
Interrupt number	1 … 16
Angle	0° … 719.9°

Periodic interrupts These are the ranges for periodic interrupts:

Parameter	Range
Interrupt number	Periodic 1 … 6
Start angle	0° … 719.9°
Angle interval	0.1° … 720°

Related topics

Migrating Engine Control from MicroAutoBox II to RapidPro

Objective

MicroAutoBox II applications can be modified so that they can be executed on MicroAutoBox II devices that use a RapidPro system as an I/O subsystem.

Tip

You can regard the `DemoRPCUEngineControl.mdl` demo model (RapidPro) as the migrated version of the `Demo1401ExtendedEngineCtrl.mdl` demo model (MicroAutoBox II).

Where to go from here

Information in this section

Engine Control: RapidPro Versus MicroAutoBox II

MicroAutoBox II variants with extended engine control

The following MicroAutoBox II variants support engine control up to dSPACE release 2018-B:

- MicroAutoBox II 1401/1501
- MicroAutoBox II 1401/1504
- MicroAutoBox II 1401/1505/1507

Main differences

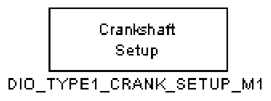
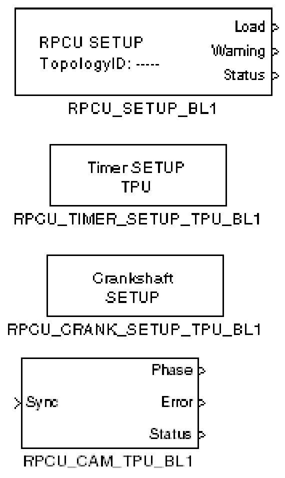
If you want to transform an engine control model running on a MicroAutoBox II into an engine control model running on a RapidPro I/O subsystem, you should be aware of the following functional differences between MicroAutoBox II and RapidPro:

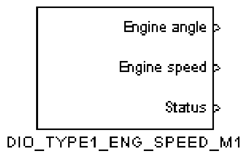
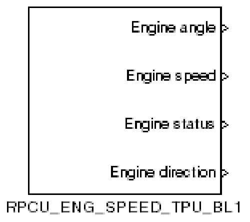
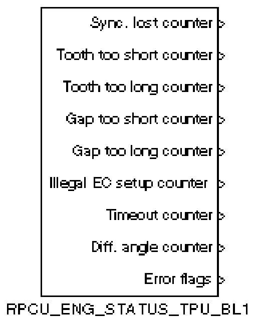
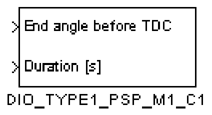
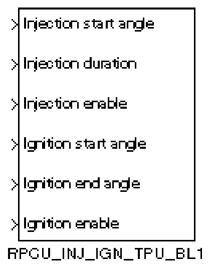
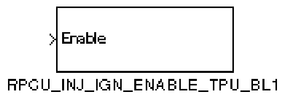
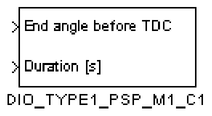
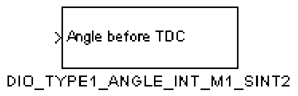
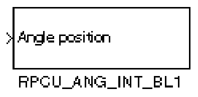
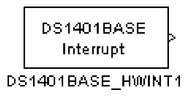
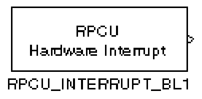
Engine Control Issue	MicroAutoBox II	RapidPro
Crankshaft signal type (active or passive)	Requires appropriate setting of the Boot mode parameter (refer to Parameters Page (DIO_TYPE1_CRANK_SETUP_Mx) (📖 MicroAutoBox RTI Reference))	Requires appropriate SC module (refer to Crankshaft Signal Reference Data on page 271)
0° position	Rising edge of the last (missing) tooth in the gap (refer to Detecting Crankshaft Wheel Gaps (MicroAutoBox Features))	Rising or falling edge of the first tooth after the gap (refer to Basics on Camshaft Sensor Signals on page 145)
Angle counter	Incrementation on every rising edge (refer to Detecting Crankshaft Wheel Gaps (📖 MicroAutoBox Features))	Incrementation on every 0.1°, repeatedly invoked on every rising edge or on every falling edge (refer to Parameters Page (RPCU_CRANK_SETUP_TPU_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖))
Resolution of angle measurement	Depends on the number of crankshaft teeth (refer to Extended Period Measurement (PMMX) for EEC (📖 MicroAutoBox Features))	Always 0.1° (refer to Crankshaft Signal Reference Data on page 271)
Resolution of pulse generation (injection/ignition)	200 ns (refer to Extended Period Measurement (PMMX) for EEC (📖 MicroAutoBox Features))	35.7 ns (refer to Injection and Ignition Reference Data on page 276)

For details on engine control features that come (exclusively) with the RapidPro Control Unit Blockset, refer to Engine Control on page 135.

RTI block substitution

If you want to transform an engine control model running on a MicroAutoBox II into an engine control model running on a RapidPro I/O subsystem, you must replace the MicroAutoBox II blocks with blocks of the RapidPro Control Unit blockset as follows:

MicroAutoBox II Block	Corresponding RPCU Block(s)
	 RPCU SETUP TopologyID: ----- Load Warning Status RPCU_SETUP_BL1 Timer SETUP TPU RPCU_TIMER_SETUP_TPU_BL1 Crankshaft SETUP RPCU_CRANK_SETUP_TPU_BL1 Sync Phase Error Status RPCU_CAM_TPU_BL1

MicroAutoBox II Block	Corresponding RPCU Block(s)
 Engine angle Engine speed Status DIO_TYPE1_ENG_SPEED_M1	 Engine angle Engine speed Engine status Engine direction RPCU_ENG_SPEED_TPU_BL1  Sync. lost counter Tooth too short counter Tooth too long counter Gap too short counter Gap too long counter Illegal EC setup counter Timeout counter Diff. angle counter Error flags RPCU_ENG_STATUS_TPU_BL1
Injection: 	 Injection start angle Injection duration Injection enable Ignition start angle Ignition end angle Ignition enable RPCU_INJ_IGN_TPU_BL1 
Ignition: 	
	
	

Replacement of DIO_TYPE1_CRANK_SETUP_Mx Blocks

Substituting RPCU blocks

The DIO_TYPE1_CRANK_SETUP_Mx block (MicroAutoBox II) must be replaced by the following RPCU blocks (RapidPro Control Unit):

- RPCU_SETUP_BLx
- RPCU_TIMER_SETUP_TPU_BLx
- RPCU_CRANK_SETUP_TPU_BLx
- RPCU_CAM_TPU_BLx

RPCU_SETUP

The RPCU_SETUP_BLx block specifies the basic settings of the RapidPro system and initializes master-slave communication. It also measures the CPU load of the MPC565 processor (slave).

RPCU_TIMER_SETUP_TPU

The RPCU_TIMER_SETUP_TPU_BLx block configures the timer settings of the time processor units (TPUs).

RPCU_CRANK_SETUP_TPU

When you replace a DIO_TYPE1_CRANK_SETUP_Mx block with an RPCU_CRANK_SETUP_TPU_BLx, note the following block dialog settings:

DIO_TYPE1_CRANK_SETUP_Mx Settings	Corresponding RPCU_CRANK_SETUP_TPU_BLx Settings
(only one block dialog page)	Unit Page (RPCU_CRANK_SETUP_TPU_BLx)
<<< NO COUNTERPART >>> (the input channel is fixed, you cannot select it)	Input port selection (specifies the input channel; the SC module is selected indirectly via the input port selection)
Module number (means DIO module)	Module number (means ECU module)
Boot mode (considers sensor type: active/passive)	<<< NO COUNTERPART >>> (automatically defined by the input port selection)
Number of PSP channels	<<< NO COUNTERPART >>>
Enable interrupt on every tooth	<<< NO COUNTERPART >>> (Requires implementation of a periodic interrupt, refer to RPCU_ANGLE_INT_BLx)
(only one block dialog page)	Parameters Page (RPCU_CRANK_SETUP_TPU_BLx)
Enable crankshaft sensor signal noise detection	<<< NO COUNTERPART >>>
Number of crankshaft teeth without noise detection	<<< NO COUNTERPART >>> (workaround: → increase tooth period ratio)
Lower speed limit (defines speed range/resolution)	<<< NO COUNTERPART >>> (internally set to 40 rpm)

DIO_TYPE1_CRANK_SETUP_Mx Settings	Corresponding RPCU_CRANK_SETUP_TPU_BLx Settings
Resulting engine speed range	<<< NO COUNTERPART >>> ▪ (speed range is always 40 ... 20000 rpm) ▪ (speed resolution depends on RPCU_TIMER_SETUP_TPU_BLx)
<<< NO COUNTERPART >>> (internally set to "Rising")	Edge polarity (→ choose "Rising")
<<< NO COUNTERPART >>> (internally set to 0°)	Start angle (→ must be set to 720° - 360° / Number of crankshaft teeth)
<<< NO COUNTERPART >>> (Internally set as follows:) ▪ 360° / Number of crankshaft teeth (before synchronization) ▪ 360° / Missing teeth per gap + 1 (after synchronization)	Period measurement angle

RPCU_CAM_TPU

When you replace a DIO_TYPE1_CRANK_SETUP_Mx block with an RPCU_CRANK_SETUP_TPU_BLx, you additionally need to implement an RPCU_CAM_TPU_BLx. Note the following block dialog settings:

DIO_TYPE1_CRANK_SETUP_Mx Settings	Corresponding RPCU_CAM_TPU_BLx Settings
(only one block dialog page)	Parameters Page (RPCU_CAM_TPU_BLx)
Vector of camshaft synchronization start angles	Vector of camshaft start angles
Vector of camshaft synchronization end angles	Vector of camshaft end angles
<<< NO COUNTERPART >>> (input port is determined by the system)	Input port selection (→ select a camshaft wheel)
<<< NO COUNTERPART >>> (system checks whether the camshaft signal is HIGH between camshaft synchronization start angle and camshaft synchronization end angle)	Edge polarity (system checks the camshaft signal for relevant edges between segment start angle and segment end angle)
<<< NO COUNTERPART >>>	Camshaft tolerance (considers production tolerances of the camshaft wheel)
<<< NO COUNTERPART >>>	Signal polarity before first transition (→ must be set to "Low")
<<< NO COUNTERPART >>> (internally set to 0° ... 720°)	Segment start/end angle (segmentation decreases worst-case synchronization time – useful for camshaft wheels with multiple cams)

Replacement of DIO_TYPE1_ENG_SPEED_Mx Blocks

Substituting RPCU blocks

The DIO_TYPE1_ENG_SPEED_Mx block (MicroAutoBox II) must be replaced by the following RPCU blocks (RapidPro Control Unit):

- RPCU_ENG_SPEED_TPU_BLx
- RPCU_ENG_STATUS_TPU_BLx

RPCU_ENG_SPEED_TPU

When you replace a DIO_TYPE1_ENG_SPEED_Mx block with an RPCU_ENG_SPEED_TPU_BLx, the block output ports correspond as follows:

DIO_TYPE1_ENG_SPEED_Mx Outports	**Corresponding RPCU_ENG_SPEED_TPU_BLx Outports**
<<< NO COUNTERPART >>>	Engine status (outputs the current phase of the synchronization process)
Status (outputs an error code)	<<< NO COUNTERPART >>>
<<< NO COUNTERPART >>>	Status (optional output port) (outputs a flag indicating whether block output has been updated since last read access)

RPCU_ENG_STATUS_TPU

When you replace a DIO_TYPE1_ENG_SPEED_Mx block with an RPCU_ENG_SPEED_TPU_BLx, you can additionally implement an RPCU_ENG_STATUS_TPU_BLx block which lets you monitor the occurrence of synchronization failures.

Tip

This block should be part of a task that is not that dominant, for example, only executed every 100 ms.

DIO_TYPE1_ENG_SPEED_Mx Outports	**Corresponding RPCU_ENG_STATUS_TPU_BLx Outports**
Status (outputs an error code)	Sync. lost counter
	Tooth too short counter
	Tooth too long counter
	Gap too short counter
	Gap too long counter
	Illegal EC setup counter
	Timeout counter
	Diff. angle counter
	Error flags
<<< NO COUNTERPART >>>	Status (optional output port) (outputs a flag indicating whether block output has been updated since last read access)

Replacement of DIO_TYPE1_ANGLE_INT_Mx_SINTy Blocks

Substituting RPCU block

The DIO_TYPE1_ANGLE_INT_Mx_SINTy block (MicroAutoBox II) must be replaced by the following RPCU block (RapidPro Control Unit):

- RPCU_ANGLE_INT_BLx

RPCU_ANGLE_INT

When you replace a DIO_TYPE1_ANGLE_INT_Mx_SINTy block with an RPCU_ANGLE_INT_BLx, the block outports correspond as follows:

DIO_TYPE1_ANGLE_INT_Mx_SINTy Outports	**Corresponding RPCU_ANGLE_INT_BLx Outports**
Angle before TDC (angle value relative to TDC)	Angle position (absolute angle value)

The dialog settings correspond as follows:

DIO_TYPE1_ANGLE_INT_Mx_SINTy Settings	**Corresponding RPCU_ANGLE_INT_BLx Settings**
(only one block dialog page)	Parameters Page (RPCU_ANGLE_INT_BLx)
Initial interrupt angle before TDC (angle value relative to TDC)	Angle (absolute angle value)

Tip

The RPCU_ANGLE_INT_BLx block dialog comprises additional settings, for example, to specify periodic interrupts.

Replacement of DS1401BASE_HWINT_Mx Blocks

Substituting RPCU block

The DS1401BASE_HWINT block (MicroAutoBox II) must be replaced by the following RPCU block (RapidPro Control Unit):

- RPCU_INTERRUPT_BLx

RPCU_INTERRUPT

When you replace a DS1401BASE_HWINT block with an RPCU_INTERRUPT_BLx, the block outports correspond as follows:

DS1401BASE_HWINT Outports	Corresponding RPCU_INTERRUPT_BLx Outports
<No label> Trigger output (Data type: Function call)	<No label> Trigger output (Data type: Function call)

The dialog settings correspond as follows:

DS1401BASE_HWINT Settings	Corresponding RPCU_INTERRUPT_BLx Settings
(only one block dialog page)	Unit Page (RPCU_INTERRUPT)
Interrupt number (indirect selection of the interrupt source via ID)	Interrupt selection (direct selection of the interrupt source)

Replacement of DIO_TYPE1_PSP_Mx_Cy Blocks

Substituting RPCU blocks

The DIO_TYPE1_PSP_Mx_Cy block (MicroAutoBox II) must be replaced by the following RPCU blocks (RapidPro Control Unit):

- RPCU_INJ_IGN_TPU_BLx
- RPCU_INJ_IGN_ENABLE_TPU_BLx

RPCU_INJ_IGN_TPU

When you replace a DIO_TYPE1_PSP_Mx_Cy block with an RPCU_INJ_IGN_TPU_BLx, the block outports correspond as follows:

Note

Using one DIO_TYPE1_PSP_Mx_Cy block, you can specify either one injection or one ignition pulse pattern. Using one RPCU_INJ_IGN_TPU_BLx block, you can specify both one injection and one ignition pulse pattern.

DIO_TYPE1_PSP_Mx_Cy Outports	**Corresponding RPCU_INJ_IGN_TPU_BLx Outports**	
Start/end angle before TDC[1]	*Injection:*	*Ignition:*
	Injection start angle	Ignition start angle
Duration	Injection duration	Ignition end angle

[1] End or start angle: Depends on the PSP mode

The dialog settings correspond as follows.

DIO_TYPE1_PSP_Mx_Cy Settings		**Corresponding RPCU_INJ_IGN_TPU_BLx Settings**
(only one block dialog page)		Unit Page (RPCU_INJ_IGN_TPU_BLx)
Module number (means DIO module)		Module number (means ECU module)
PSP mode (relates either to injection or ignition)		<<< NO COUNTERPART >>> (see the specific dialog pages of the block)
<<< NO COUNTERPART >>>		Cylinder number
Top dead center (TDC)		Top dead center angle
(only one block dialog page)		Injection Page (RPCU_INJ_IGN_TPU_BLx) and Injection Pulses Page (RPCU_INJ_IGN_TPU_BLx)
	<<< NO COUNTERPART >>>	Injection enable
	Channel	Output driver selection (specifies the output channel, the SC/PS module is selected indirectly via the output driver selection)
	Pulses per cycle	Number of pulses
	Initial angle before TDC[1]	Initial start angle before TDC
	Initial pulse duration	Initial duration
	Termination state	Termination state
	Termination angle before TDC[1]	Start angle on termination before TDC
	Termination pulse duration	Duration on termination

DIO_TYPE1_PSP_Mx_Cy Settings		Corresponding RPCU_INJ_IGN_TPU_BLx Settings
(2nd DIO_TYPE1_PSP_Mx_Cy block required)		Ignition Page (RPCU_INJ_IGN_TPU_BLx) and Ignition Pulses Page (RPCU_INJ_IGN_TPU_BLx)
	<<< NO COUNTERPART >>>	Ignition enable
	Channel	Output driver selection (specifies the output channel, the SC/PS module is selected indirectly via the output driver selection)
	Pulses per cycle	Number of pulses
	Initial angle before TDC[1)]	Initial start angle before TDC
	Initial pulse duration	Initial end angle before TDC
	Termination state	Termination state
	Termination angle before TDC[1)]	Start angle before TDC on termination
	Termination pulse duration	End angle before TDC on termination

[1)] End or start angle: Depends on the PSP mode

RPCU_INJIGN_ENABLE_TPU

The RPCU_INJ_IGN_ENABLE_TPU_BLx block is required by the RPCU_INJ_IGN_TPU_BLx. It is used to switch injection and ignition on/off for all cylinders centrally during run-time.

Tip

There is no counterpart block available in the MicroAutoBox II blockset. For switching injection and ignition off in MicroAutoBox II applications, as a workaround, you must set the pulse duration to zero.

Extended Engine Control

Objective

dSPACE offers RapidPro SC and PS modules for handling special engine data with the RapidPro system. You cannot use the specific extended engine control features without these modules.

Where to go from here

Information in this section

Information in other sections

Extended Engine Control (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Gives information about the RTI blocks concerned with extended engine control features.

Measuring Exhaust Gas Oxygen with the SC-EGOS 2/1 Module

Where to go from here

Information in this section

Information in other sections

Measuring Exhaust Gas Oxygen with the SC-EGOS 2/1 Module (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Gives information about the RTI blocks concerned with the measurement of the oxygen fraction λ of the exhaust gas and the temperature of the lambda probe.

Basics of Evaluating EGOS Signals

Signal generation

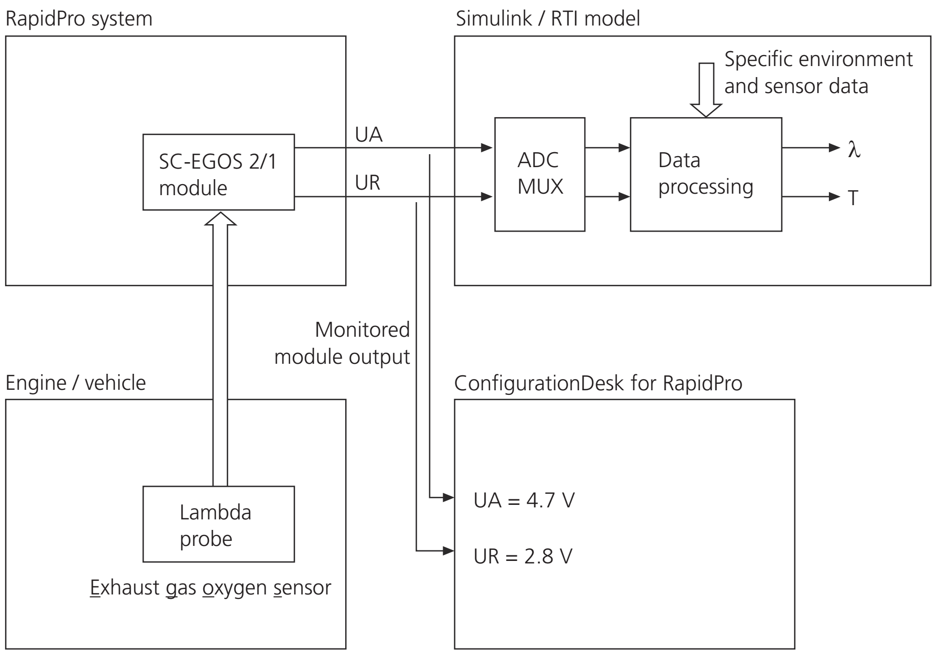
The SC-EGOS 2/1 module outputs two signals, EGOS_UA_OUT (short-term UA) and EGOS_UR_OUT (UR). UA depends on the oxygen fraction λ of the exhaust gas, UR depends on the temperature of the connected lambda probe.

Note

The signal terms in this documentation, for example, UA, correspond to the signal terms of the supported lambda probes and the Bosch CJ125 ICs. The Bosch CJ125 ICs are part of the module's hardware and control the connected lambda probes.

Signal processing

At least one LSU4.2 or LSU4.9 Bosch lambda probe must be connected to the SC-EGOS 2/1 module as part of your RapidPro system. You can use ConfigurationDesk for RapidPro for monitoring the output signals UA and UR. AD conversion and data processing via an Simulink/RTI model is needed to calculate the oxygen fraction λ and the probe's temperature T as functions of UA and UR.



For details on the required equipment, refer to SC-EGOS 2/1 Module (RapidPro System Hardware Reference 🕮).

Monitoring the output signals of the module

Monitoring the output signals UA and UR of the SC-EGOS 2/1 module enables you to check whether the system works properly, and you can roughly estimate changes of the oxygen fraction λ and the probe's temperature T.

If ConfigurationDesk for RapidPro is in online mode, UA and UR are displayed as digital numbers with one fractional digit, for example, 4.7 V. The sample rate is 200 ms. The physical unit is volt.

Calculating λ

UA is a function of the oxygen fraction λ of the exhaust gas. Knowing UA enables you to compute the pump current IP of the lambda probe, and IP lets you make a rough estimate of λ:

- IP = 0 => λ = 1
- IP < 0 => λ < 1 (rich exhaust gas)
- IP > 0 => λ > 1 (lean exhaust gas)

For details on the relation UA-IP, refer to Determining the Pump Current of a Lambda Probe (RapidPro System Hardware Reference 🕮).

For the precise determination of λ, refer to the technical documentation of the lambda probe.

Calculating T

UR is a function of the internal resistance Ri of the lambda probe. Knowing UR enables you to compute Ri, and Ri lets you determine the probe's temperature T.

For details on the relation UR-Ri, refer to Determining the Internal Resistance of a Lambda Probe (RapidPro System Hardware Reference 🕮).

For the relation between Ri and the probe's temperature T, refer to the technical documentation of the lambda probe.

Related topics

HowTos

How to Make EGOS Signals Available in the Simulink/RTI Model

Objective

Before the output signals of the SC-EGOS 2/1 module can be used to calculate the oxygen fraction λ and the temperature of the lambda probe, they must be prepared in a specific way.

Preconditions

The following preconditions must be fulfilled:

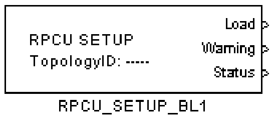
- At least one LSU4.2 or LSU4.9 Bosch lambda probe is connected to the SC-EGOS 2/1 module (RapidPro system), refer to Notes on Connecting Lambda Probes in *SC-EGOS 2/1 Module* of the RapidPro System Hardware Reference 🕮.
- The EGOS channel has been configured, that is, the measurement range for λ and the pump reference current IP_{Ref}, refer to Configuration via ConfigurationDesk for RapidPro in *SC-EGOS 2/1 Module* of the RapidPro System Hardware Reference 🕮.

Method

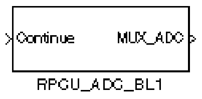
To make EGOS signals available in the Simulink/RTI model

1 Open a new Simulink model.

2 From the **rtirpcu** library, copy an RPCU_SETUP_BLx block to the model to load the hardware topology file for RTI.



3 From the **rtirpcu** library, copy an RPCU_ADC_BLx block to the model (AD converter). The block is required for digitizing analog voltage signals UA and UR.



4 Double-click the RPCU_ADC_BLx block to open its dialog and specify its settings:
   - The settings on the **Unit** page depend on the connected RCP system.
   - The available EGOS channels are displayed on the **Parameters** page.

5 Click **OK** to close the dialog.
   If there are wrong settings or conflicts with other channels, an error message will appear.

Result

EGOS signals UA and UR are available at the MUX ADC outport of the RPCU_ADC_BLx block for data processing, for example, to calculate λ and the probe's temperature T.

For the precise calculation of λ and the probe's temperature T, refer to the technical documentation of the connected lambda probe(s).

Related topics

Basics

Basics of Evaluating EGOS Signals..294

Diagnostics for the SC-EGOS 2/1 Module

Diagnostic messages in ConfigurationDesk for RapidPro

If ConfigurationDesk for RapidPro is in online mode, you get comprehensive diagnostic feedback on the status of the SC-EGOS 2/1 module and the status of the connection to external devices.

Detection of short circuits

For example, each time a short circuit is detected on the I/O circuit of the SC-EGOS 2/1 module, a specific diagnostic message is displayed in ConfigurationDesk for RapidPro, and the channel concerned is shut down.

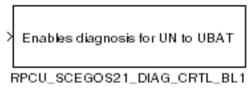
However, a short circuit from the module's front connector pin UN-to-UBAT might be detected erroneously, if the internal resistance Ri of the lambda probe is high (cold probe), and you have set the pump reference current to IP = 0 A in ConfigurationDesk for RapidPro. To prevent this, you can disable this detection service as long as the connected lambda probe is cold, refer to How to Implement Run-Time Control of UN-to-UBAT Short Circuit Detection on page 299.

For details on diagnostic messages, refer to Diagnostics in *SC-EGOS 2/1 Module* of the RapidPro System Hardware Reference 🕮 .

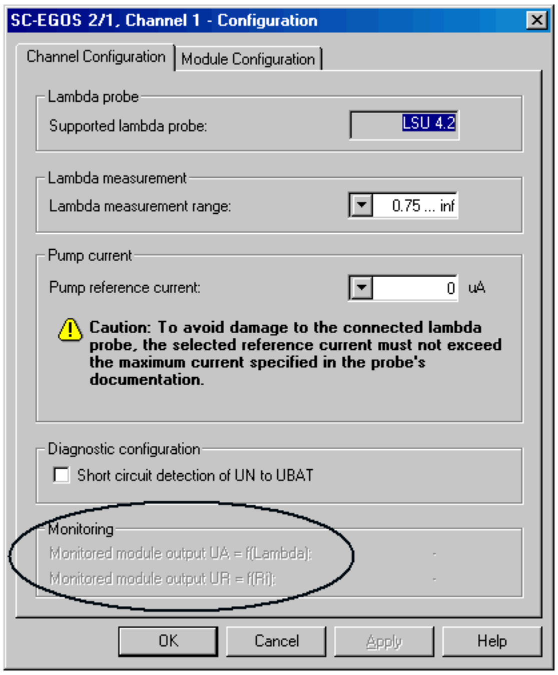
Enabling short circuit detection

You can enable/disable the short circuit detection from the module's front connector pin UN-to-UBAT at run-time and outside run-time. The setting chosen at run-time overwrites the setting chosen outside run-time.

1. At run-time (prio 1): Via RPCU_SCEGOS21_DIAG_CTRL_BLx block, refer to How to Implement Run-Time Control of UN-to-UBAT Short Circuit Detection on page 299.



2. Outside run-time (prio 2): Via ConfigurationDesk for RapidPro (Channel Configuration page).



Lambda probe
Supported lambda probe: LSU 4.2

Lambda measurement
Lambda measurement range: 0.75 ... inf

Pump current
Pump reference current: 0 uA

Caution: To avoid damage to the connected lambda probe, the selected reference current must not exceed the maximum current specified in the probe's documentation.

Diagnostic configuration
☐ Short circuit detection of UN to UBAT

Monitoring
Monitored module output UA = f(Lambda): -
Monitored module output UR = f(Ri): -

Note

If an application using an RPCU_SCEGOS21_DIAG_CTRL_BLx block stops, the setting for the short circuit detection chosen via ConfigurationDesk for RapidPro becomes the active one again.
All the settings that you can specify in ConfigurationDesk for RapidPro are stored on the module.

Related topics

HowTos

How to Implement Run-Time Control of UN-to-UBAT Short Circuit Detection

Objective

To prevent erroneous detection of a short circuit from the module's front connector pin UN-to-UBAT (due to a lambda probe being too cold, and the pump reference current configured to IP = 0 A in ConfigurationDesk for RapidPro), you can enable this specific short circuit detection service at run-time as soon as the temperature of the connected lambda probe is high enough.

Restrictions

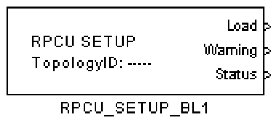
A second RPCU_SCEGOS21_DIAG_CRTL_BLx block with the same Channel, Slot, and Layer numbers is not allowed in the model.

Method

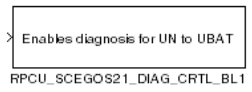
To implement run-time control of UN-to-UBAT short circuit detection

1 Open a new Simulink model.

2 From the rtirpcu library, copy an RPCU_SETUP_BLx block to the model to load the hardware topology file for RTI.



3 From the rtirpcu_scegos21_lib sublibrary, copy an RPCU_SCEGOS21_DIAG_CTRL_BLx block to the model.



4 Connect the Enables diagnosis for UN-to-UBAT inport of the block to a Boolean variable which controls the short circuit detection from UN-to-UBAT at run-time:

0: Detection is disabled

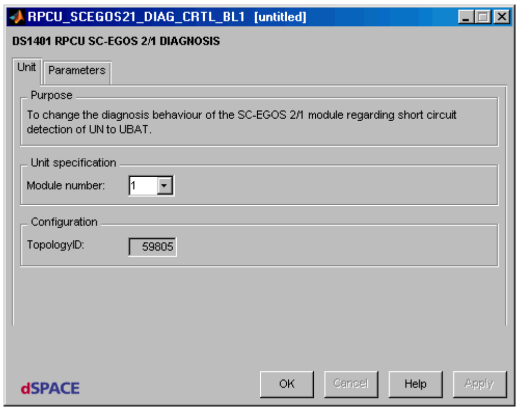
1: Detection is enabled

Note

The signal at the Enables diagnosis for UN-to-UBAT inport overwrites the Short circuit detection of UN-to-UBAT option on the Channel Configuration page in ConfigurationDesk for RapidPro.
Other short circuit detection services of the SC-EGOS 2/1 module are not affected by the RPCU_SCEGOS21_DIAG_CTRL_BLx block.

5 Double-click the block to open its dialog.

The Unit page is displayed.



RPCU_SCEGOS21_DIAG_CRTL_BL1 [untitled]

DS1401 RPCU SC-EGOS 2/1 DIAGNOSIS

Unit | Parameters

Purpose
To change the diagnosis behaviour of the SC-EGOS 2/1 module regarding short circuit detection of UN to UBAT.

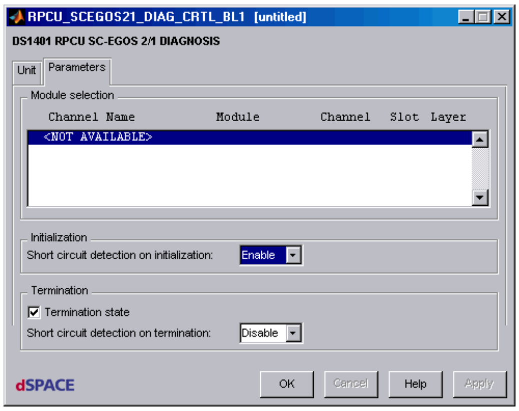
Unit specification
Module number: 1

Configuration
TopologyID: 59805

6 In the Board/Module number list, choose the same board/module number as for the RPCU_SETUP Block.

7 In the ECU channel list, choose the same ECU channel number as for the RPCU_SETUP Block. (Only available if you work with a DS1007 modular system)

8 Change to the Parameters page.



RPCU_SCEGOS21_DIAG_CRTL_BL1 [untitled]

DS1401 RPCU SC-EGOS 2/1 DIAGNOSIS

Unit | Parameters

Module selection

```
Channel Name          Module          Channel  Slot  Layer
<NOT AVAILABLE>
```

Initialization
Short circuit detection on initialization: Enable

Termination
☑ Termination state
Short circuit detection on termination: Disable

9 Select the desired EGOS module in the **Module selection** list.

> **Note**
>
> A second RPCU_SCEGOS21_DIAG_CRTL_BLx block with the same channel, slot, and layer numbers is not allowed in the model.

10 Specify the behavior of short circuit detection on initialization and termination.

Result

You have implemented run-time control of UN-to-UBAT short circuit detection.

Related topics

Basics

References

RPCU_SCEGOS21_DIAG_CTRL_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Measuring Exhaust Gas Oxygen with the SC-UHEGO 2/1 Module

Objective

You can measure the exhaust gas oxygen of combustion engines in run time by using up to two lambda probes from Denso connected to the SC-UHEGO 2/1 module from dSPACE (UHEGO means Universal Heated Exhaust Gas Oxygen).

Required knowledge

- Complete knowledge on the connected lambda probes is recommended for working with the SC-UHEGO 2/1 module. Refer to the technical documentation of the specific lambda probe.
- The signal terms in this section correspond to the signal terms of the supported lambda probes and the Denso ICs on the module. The Denso ICs are part of the module's hardware and control the connected lambda probes.

For hardware-related reference information on the SC-UHEGO 2/1 module, refer to SC-UHEGO 2/1 Module (RapidPro System Hardware Reference 🕮).

Where to go from here

Information in this section

Basics on Measuring Exhaust Gas Oxygen

Where to go from here

Information in this section

Basics on Lambda Sensors

Field of application

A lambda sensor measures the remaining oxygen content in the exhaust gas of a combustion engine so that the engine ECU can determine the amount of fuel required to ensure optimal exhaust gas values. The remaining oxygen content is expressed by the variable λ (Greek: lambda):

$$\lambda = \frac{\text{Measured air-to-fuel ratio}}{\text{Stoichiometric air-to-fuel ratio}} = \frac{\text{Measured air-to-fuel ratio}}{14.7 : 1}$$

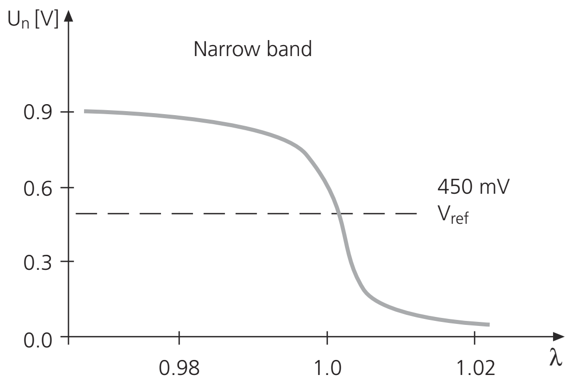
Combustion is best if the air/fuel ratio (AFR) is 14.7 : 1, thus $\lambda = 1$ (stoichiometric AFR).

- $\lambda < 1.0$: Rich air/fuel mixture
- $\lambda > 1.0$: Lean air/fuel mixture

Narrowband lambda sensors

Narrowband lambda sensors were the first design type of lambda sensors. Their operating principle is as follows:

The O_2 concentration is measured in the Nernst cell, which generates a voltage signal depending on the O_2 concentration. When the air/fuel ratio is perfectly balanced ($\lambda = 1$), a narrowband sensor generates a voltage signal of about 450 mV. When the fuel mixture goes rich, even just a little, the sensor's voltage output shoots up to its maximum output of about 900 mV. Conversely, when the fuel mixture goes lean, the sensor's output voltage drops to 100 mV. The reaction time for a lean/rich or rich/lean voltage jump of 0.2 V ↔ 0.8 V is approx. 300 ms. The reaction time of the Nernst cell at working temperature is at maximum 50 ms.



Every time the voltage signal jumps or drops, the ECU responds by decreasing or increasing the amount of fuel that is delivered. This rapid flip-flopping back and forth allows the feedback fuel control system to maintain a more-or-less balanced air-fuel mixture, on average. Narrowband sensors indicate only a rich/lean state, so they are also known as step-change sensors or binary sensors. Because they cannot measure the exact air/fuel ratio, they are not accurate enough to meet the latest emissions requirements.

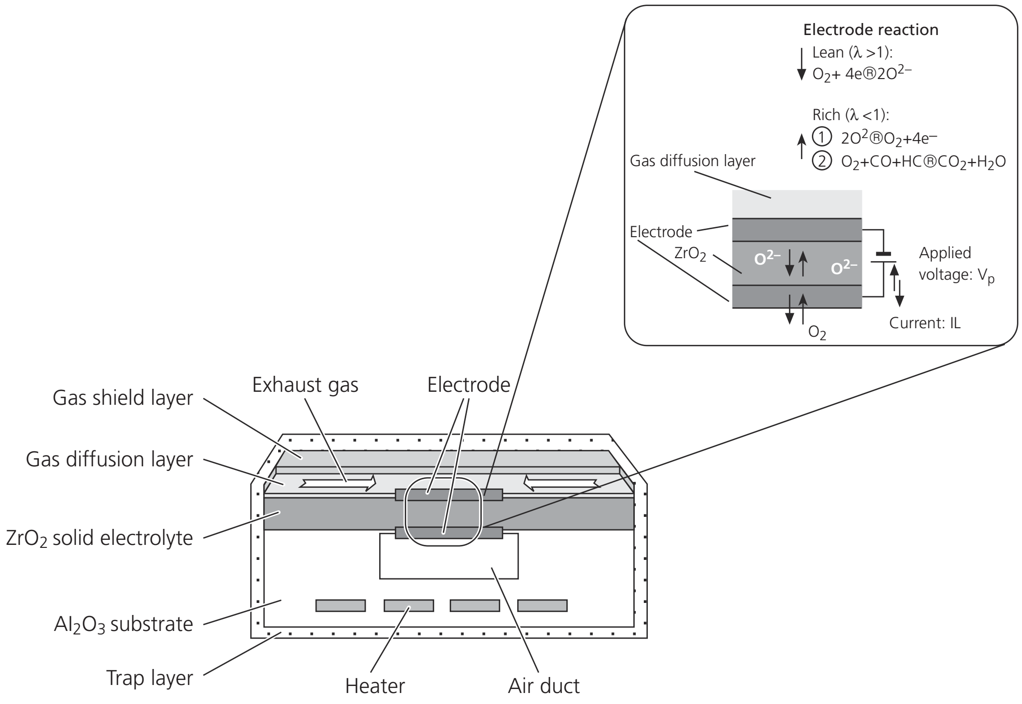
Tip

Narrowband lambda sensors are useful for tuning an engine for steady state cruising.

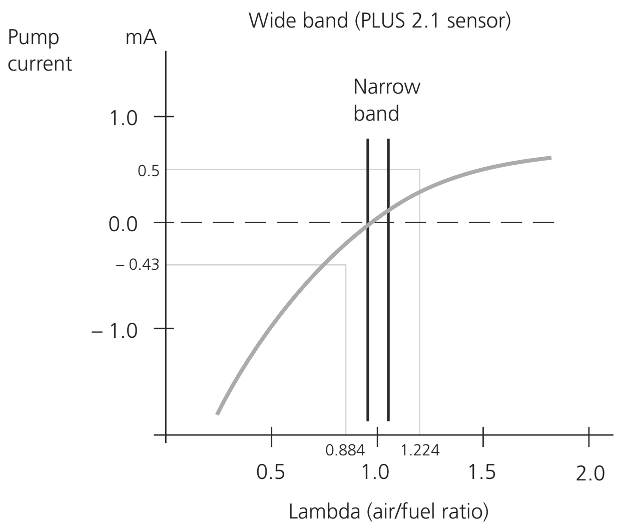
Broadband lambda sensors

Broadband lambda sensors are advanced of lambda sensors. Their operating principle is as follows:

Like narrowband lambda sensors, broadband lambda sensors have a Nernst cell and also an additional second cell. In lambda sensors from Denso, for example, this cell consists of a zirconia solid electrolyte (ZrO_2) integrated with an aluminium heating substrate (Al_2O_3). On the upper side, exhaust gas is directed through the cell and along the electrode, while on the lower side, reference air is directed along the other electrode. A specific voltage V_p is applied between the two electrodes. When the concentration of oxygen in the exhaust gas is different from that in the reference air, O_2 diffusion through the zirconia solid starts and a λ-specific pump current IL flows.



For rich air/fuel mixtures ($\lambda < 1.0$) the pump current is negative, for lean air/fuel mixtures ($\lambda > 1.0$) it is positive. The pump current increases strictly. The graph below applies to PLUS 2.1 lambda sensors from Denso.



This sensor type eliminates the lean-rich cycling inherent in narrowband sensors, allowing the control unit to adjust the fuel delivery and ignition timing of the engine much more rapidly. In the automotive industry this sensor is also called a UEGO (Universal Exhaust Gas Oxygen) sensor.

Tip

Broadband lambda sensors are also useful for tuning an engine for quick changes of loads and/or revolutions.

PLUS2.1 and PLUS3.x lambda sensors

The PLUS2.1 and PLUS3.x lambda sensors from Denso are broadband sensors. They provide two possible air/fuel ratio (AFR) ranges (software-configurable):

- Wide AFR range: 10 … ∞ (λ = 0.68 ... ∞)
- Stoichiometric AFR range: 12 … 25 (λ = 0.82 ... 1.7)

Because the PLUS2.1 or PLUS3.x lambda sensors have internal heaters to reach the working temperature (700 °C, 1292 °F) quickly, they are called UHEGO (Universal Heated Exhaust Gas Oxygen) sensors.

The PLUS2.1 or PLUS3.x lambda sensors have four connector pins:

- Two pins for the positive and negative sensor output (must be connected to the SC-UHEGO 2/1 module)
- Two pins for the internal heater (must be connected to a power stage module, e.g., PS-LSD 6/1 module)

For a more detailed description of the PLUS2.1 or PLUS3.x lambda sensors, refer to the technical documentation provided by DENSO.

Basics on Measuring the Exhaust Gas Oxygen with the SC-UHEGO 2/1 Module

Signal conditioning

You can use the SC-UHEGO 2/1 module as part of a RapidPro system to connect PLUS2.1 or PLUS3.x lambda probes. Up to two sensors can be connected to one module. The module has the following main characteristics:

Probe type specification The module provides two channels to connect either a PLUS2.1 or PLUS3.x lambda sensor (software-configurable per probe).

Note

You cannot use lambda sensor other than the PLUS2.1 or PLUS3.x sensor with the SC-UHEGO 2/1 module.

Required software ConfigurationDesk for RapidPro with plug-in software (dSPACE Release 5.4...6.2), or ConfigurationDesk for RapidPro as of Release 6.3 (without plug-in software) is required for working with the SC-UHEGO 2/1 module. If the SC-UHEGO 2/1 module is installed in a Control Unit or a stack with unit connection bus (UCB), you must use the RapidPro Control Unit Blockset to implement the measurement.

Output signals Each of the two channels of the SC-UHEGO 2/1 module (one channel per lambda sensor) provides the following output signals:

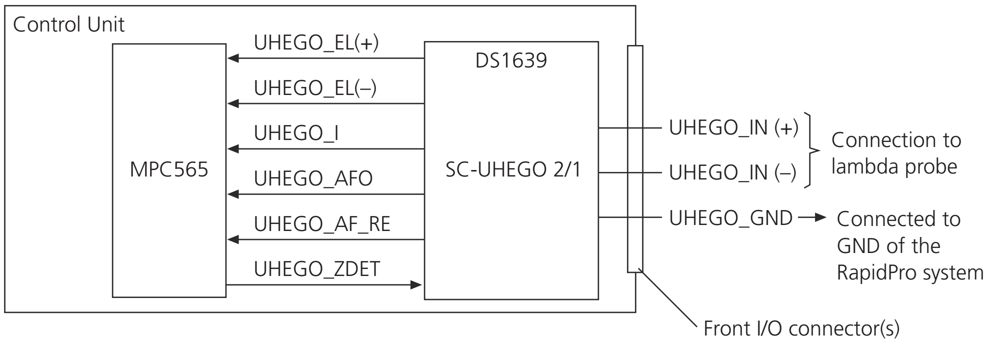
Output Signal[1]	**Used to ...**
UHEGO_EL(+)[2]	Measure air/fuel ratio (AFR) → Calculation of λ
UHEGO_AFO	
UHEGO_I[3]	Measure internal resistance of the lambda probe → Calculation of the probe's temperature
UHEGO_AF_RE	Measure validity of the UHEGO_AFO signal[4]

[1] The signal terms correspond to the signal terms of the supported lambda probes and the Denso ICs on the module.
[2] UHEGO_EL(-) is not needed for measurement. You can use this signal for enhanced diagnostic purposes.
[3] Needs to be triggered by UHEGO_ZDET input signal.
[4] UHEGO_AFO and UHEGO_I cannot not be measured at the same time.

Tip

The following sample rates are appropriate:
- UHEGO_AFO: 10 ms
- UHEGO_I and UHEGO_EL(+): 60…100 ms

The following illustration gives a simplified connection example:



For details on I/O connections, refer to Notes on Driving Denso Lambda Probes (RapidPro System Hardware Reference 📖).

Input signals The SC-UHEGO 2/1 module requires the following digital input signal for each lambda probe (see the illustration for the output signals above):

Input Signal	Used to ...
UHEGO_ZDET	Trigger measurement of UHEGO_I[1)]

1) If a falling edge in UHEGO_ZDET occurs, UHEGO_I is measured once.

Note

It is not possible to measure the UHEGO_AFO and UHEGO_I signals at the same time. The air/fuel ratio measurement is invalid for approx. 4.5 ms during the internal resistance measurement.

For detailed information on the module, its signals, and how to install it, refer to SC-UHEGO 2/1 Module (RapidPro System Hardware Reference 📖).

Calculation of λ

You can calculate the air/fuel ratio (AFR), and hence λ, if you know the pump current IL of the lambda probe.

Determination of the pump current IL The signals UHEGO_AFO and UHEGO_EL(+) let you determine the pump current IL of the lambda probe. You can calculate IL with the following formula:

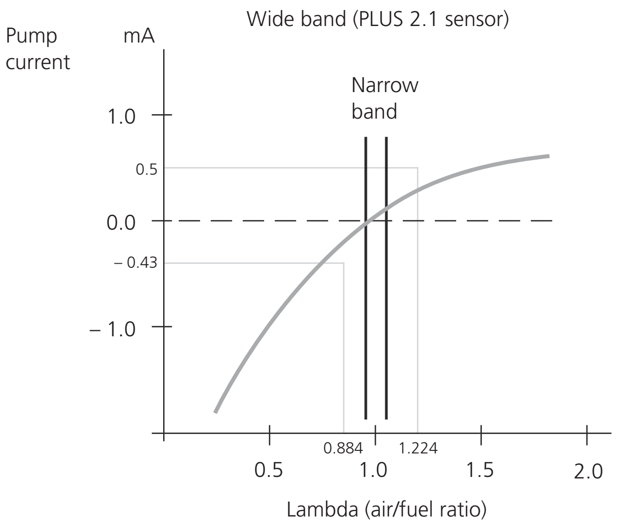
$$\text{IL [mA]} = \frac{\text{UHEGO\_AFO [V]} - \text{UHEGO\_EL(+) [V]}}{\text{Factor}_{\text{IL}}} \times 1000$$

$\text{Factor}_{\text{IL}}$ in the formula above depends on the setting of the measurement range and on the connected lambda probe as shown in the table below:

Parameter	Lambda Probe	$\text{Factor}_{\text{IL}}$
Wide range	PLUS2.1	720
	PLUS3.x	1200

Parameter	**Lambda Probe**	**Factor$_{IL}$**
Stoichiometric range	PLUS2.1	1380
	PLUS3.x	2299.6

Calculation of λ λ is a function of the pump current IL, and this relationship is probe-specific. The example below is the IL-λ-curve for the PLUS2.1 lambda probe (graphic from the probe's technical documentation):



For detailed information on the relation between AFR and the current IL, refer to the technical documentation of the specific lambda probe.

Note

You get a valid measurement of lambda values, if the value of R_{ac} is in a valid range and the UHEGO_AF_RE signal indicates the termination of the measurement.

Calculation of the probe's temperature

The temperature of a lambda probe can be deduced from its internal resistance.

Determination of the internal resistance R_{ac} The signal UHEGO_I lets you determine the internal resistance R_{ac} of the lambda probe. You can calculate R_{ac} with the following formula:

$$R_{ac}\,[\Omega] = \frac{\text{Factor}_{ac}}{\text{UHEGO\_I}\,[V]} \times 0.4$$

Anticipated range of R_{ac}: 16 Ω … ∞

$Factor_{ac}$ in the formula above depends on the setting of the measurement range and on the connected lambda probe as shown in the table below:

Parameter	**Lambda Probe**	**$Factor_{ac}$**
Wide range	PLUS2.1	120
	PLUS3.x	200
Stoichiometric range	PLUS2.1	
	PLUS3.x	

Calculation of the probe's temperature The probe's temperature is a function of the internal resistance R_{ac}, and this relationship is probe-specific. For detailed information on the connection between R_{ac} and the probe’s temperature, refer to the technical documentation of the specific probe.

Tip

You do not need to know the exact curve, but only the resistance value that corresponds to the working point of the probe.

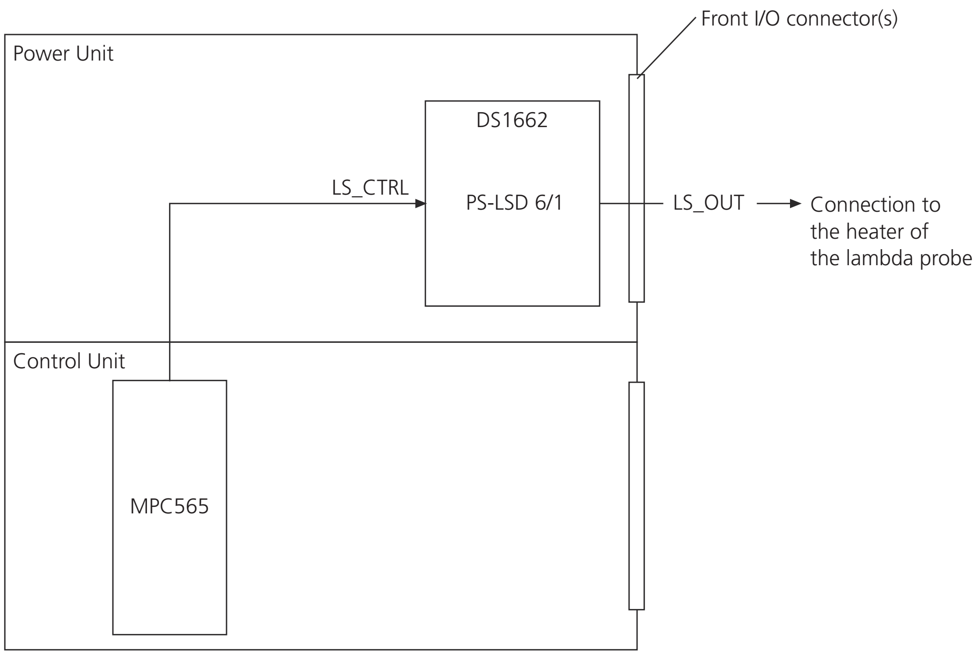
After the startup phase when the working point of the probe is achieved, that is, the probe's temperature is about 700 °C (1292 °F), R_{ac} reads as follows:

Lambda Probe	**$R_{ac, working point}$**
PLUS2.1	approx. 28 Ω
PLUS3.x	approx. 39 Ω

Driving the heater of the lambda probes

In standard use cases the heating of the lambda probes has to be done by the RapidPro system, too. This requires a RapidPro power stage module (PS module), in addition to the SC-UHEGO 2/1 module. There are a number of PS modules that are suitable for heating the lambda probes, e.g., the PS-LSD 6/1 module. The main criteria for selecting one is the maximum heat current.

The following illustration gives a simplified connection example:



For details on I/O connections, refer to Notes on Driving Denso Lambda Probes (RapidPro System Hardware Reference 📖).

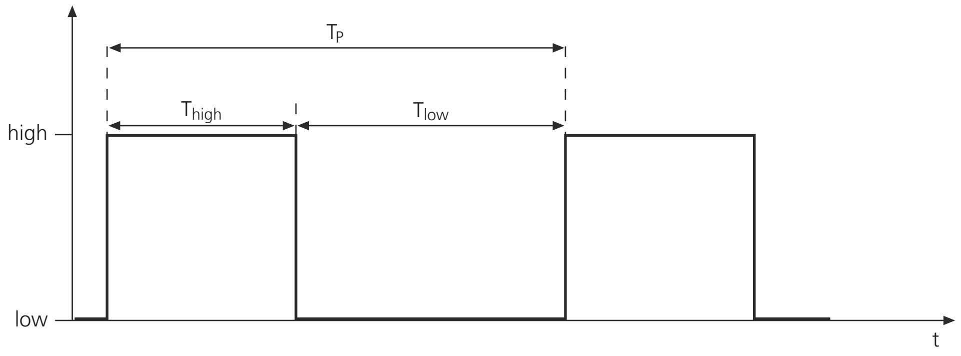
Maximum heat current The resistance of the heater is minimum at very low ambient temperatures and rises with higher temperatures. For the PLUS2.1 and PLUS3.x lambda probes, the resistance of the heater is approx. 1.4 Ω at -40 °C (-40 °F). Thus, the current maximum occurs at very low ambient temperatures at the start of operation. The current maximum of course also depends on the voltage level of the power supply.

Note

It might be necessary to limit the current at the beginning of the heating process to prevent thermal shocks. This can be done by limiting the duty cycle of the PWM signal.
If multiple channels of the PS module need to be coupled in parallel, you need custom routing code. Contact dSPACE for appropriate routing code.

For detailed information on the relation of temperature and heater resistance, refer to the technical documentation of the specific lambda probe.

Duty cycle as control signal The duty cycle of a PWM signal is defined as d = T_{high}/ T_P. The available duty cycle range is 0 … 1 (0 … 100 %).



Depending on the specific PS module, the incoming PWM signal might be inverted. The PS-LSD 6/1 module has the following relationship between duty cycle (PWM signal) and current (output):

Duty Cycle d	Heat current
0	Maximum
1	0

For the relationship between duty cycle and current of other PS modules, refer to the RapidPro System Hardware Reference 📖.

Note

For PWM-shaped currents flowing through ohmic resistances, the effective current reads:

$\text{Current}_{\text{effective}} = d * \text{Current}_{\text{maximum}}$

Current feedback measurement If it is necessary to know the current applied to the heater, the current feedback measurement of PS modules can be used. If there are two power channels switched in parallel, the current feedback must be measured on both channels, and the values must be added in the real-time application. However, current feedback measurement is only possible with the first two channels of a PS module.

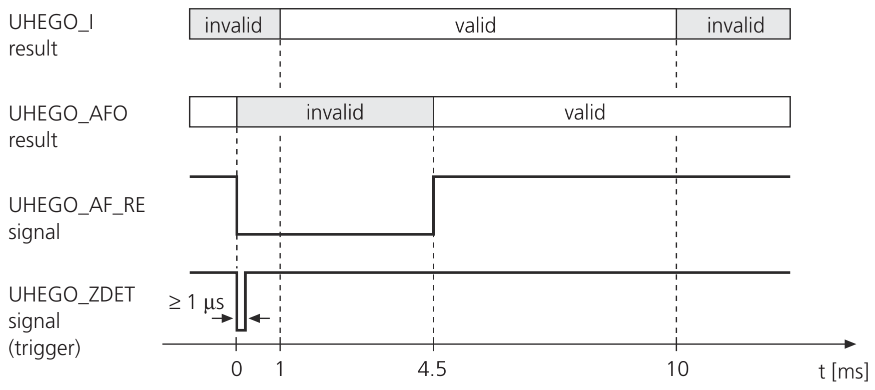
For more detailed information about RapidPro Units and PS modules, refer to the RapidPro System Hardware Reference 📖.

Measurement restrictions

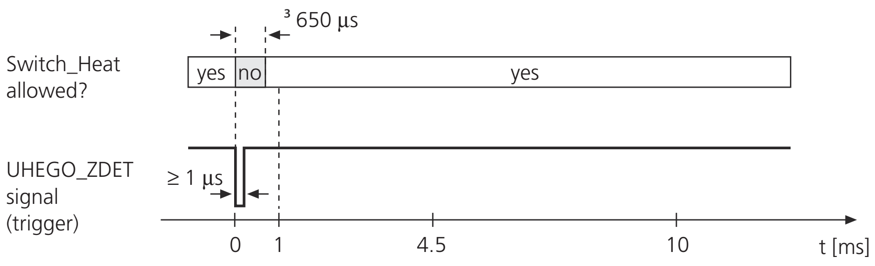
There are some restrictions when using the SC-UHEGO 2/1 module.

UHEGO_AFO and UHEGO_I measurements The measurement of UHEGO_I disturbs the measurement of UHEGO_AFO. In consequence you cannot measure UHEGO_AFO while UHEGO_I is being measured. To solve this problem, the SC-UHEGO 2/1 module provides the following signal arrangement:

- The module requires the digital input signal UHEGO_ZDET to trigger the measurement of UHEGO_I. The trigger event is the falling edge of UHEGO_ZDET. After a wait time of about 1 ms the measurement result for UHEGO_I can be read at the UHEGO_I signal outport of the module. The signal becomes invalid again 10 ms after the falling edge of UHEGO_ZDET.
- The module outputs the UHEGO_AF_RE signal. It is at a low level for a period of about 4.5 ms after the falling edge of the UHEGO_ZDET trigger signal. A low level means that the UHEGO_AFO signal is invalid.



Heat current and UHEGO_I measurement The heat current disturbs the measurement of UHEGO_I (internal resistance). This is due to the fact that the heat current has a high-frequency temporal structure (PWM). The heat current has the same high-frequency temporal structure as the PWM-shaped LS_CTRL heater control signal. Thus, the heat current must be time-invariant during the measurement of UHEGO_I (i.e., constantly maximum or zero), so the duty cycle of the PWM signal must be either 0 or 1 during this time (≥ 650 µs).



Note

To be sure that the heat current is constant during the measurement of UHEGO_I, the PWM signal should stop shortly before the falling edge of UHEGO_ZDET.

Start-up phase After a real-time application has been started on a RapidPro stack or after a Single Unit has been activated, the SC-UHEGO 2/1 module needs a start-up time of approx. 700 ms. During this time the output signals of the module are invalid.

Related topics

Basics

Determining the Current of a Lambda Probe (RapidPro System Hardware Reference 📖)
Determining the Internal Resistance of a Lambda Probe (RapidPro System Hardware Reference 📖)

References

Notes on Driving Denso Lambda Probes (RapidPro System Hardware Reference 📖)
SC-UHEGO 2/1 Module (RapidPro System Hardware Reference 📖)

Basics on Software Implementation

RPCU Control Unit Blockset To generate the input signals and to measure the output signals, the Simulink controller model must provide appropriate blocks of the RPCU Control Unit Blockset.

Signal	Required Block
UHEGO_EL(+)	RPCU_ADC
UHEGO_AFO	
UHEGO_I	
UHEGO_AF_RE	RPCU_BIT_IN_TPU – or – RPCU_BIT_IN
UHEGO_ZDET	RPCU_BIT_OUT_TPU – or – RPCU_BIT_OUT

Note

RPCU_ADC blocks have an output value range of [0…1], so scaling to the actual module voltage output range is necessary. You can calculate the voltage values for UHEGO_EL(+), UHEGO_AFO, and UHEGO_I with the following formula:

```
Voltage value [V] = ADC_VALUE x 5.0
```

For details on the blocks, refer to RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮.

Related topics

References

RapidPro System – I/O Subsystem MPC565 RTI Reference

UHEGO Demo Application

Objective

The main implementation details of the `UHEGO_example_model.mdl` real-time demo model are explained.

Requirements

Hardware The application is based on a RapidPro system that consists of the following elements:

- RapidPro stack (Control Unit with SC-UHEGO 2/1 module and Power Unit with PS-LSD 6/1 module)
- A PLUS2.1 and a PLUS3.x lambda probe

Base sample rate The base sample rate of the real-time demo model is 1 ms.

Where to go from here

Information in this section

UHEGO Demo Application: Feeding the Measurement Signals in the Model

Objective

This section explains how the analog and digital measurement signals are fed in the `UHEGO_example_model.mdl` demo model. The relevant subsystems are introduced, as well as crucial implementation aspects.

Input+Scaling subsystem

The path of this subsystem in the demo model is `UHEGO_example_model/Input+Scaling`.

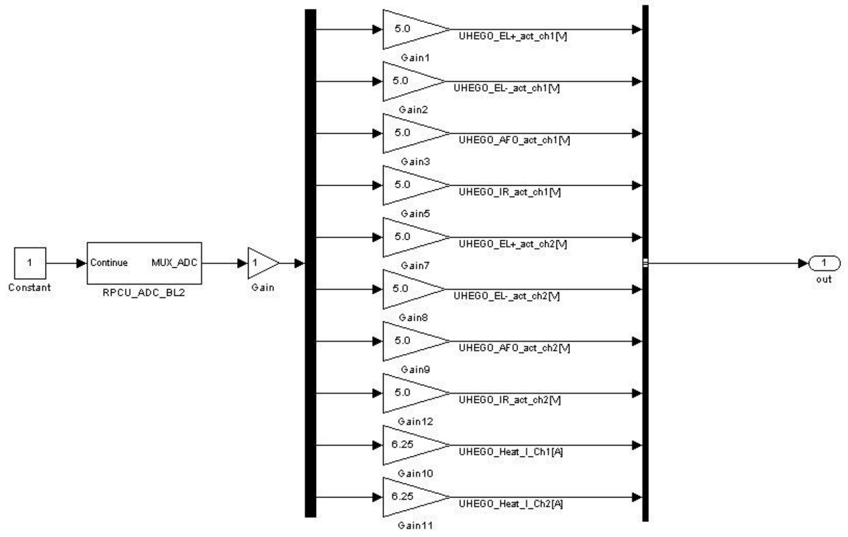
This subsystem on the root level of the model is responsible for the measurement of all the analog and digital input signals. It contains the following subsystems:

- ADC_input subsystem
- Bit_IO_input subsystem

ADC_input subsystem

The path of this subsystem in the demo model is `UHEGO_example_model/Input+Scaling/ADC_input`.

The RPCU_ADC_BL2 block is sufficient to measure the 10 analog signals that are to be processed in the real-time model. As all the signals are measured via the same block, they are all measured with the same sample time (base sample time of the model, i.e., 1 ms).



The conversion mode of the RPCU_ADC_BLx is set to Continue, so the block has a Continue inport.

These are the voltage output signals of the SC-UHEGO 2/1 module which are to be processed in the real-time model (requires eight A/D channels):

- UHEGO_EL+_act_chx [V] (requires a gain of 5.0)
- UHEGO_EL-_act_chx [V] (requires a gain of 5.0)

- UHEGO_AFO_act_chx [V] (requires a gain of 5.0)
- UHEGO_IR_act_chx [V] (requires a gain of 5.0)

Note

RPCU_ADC blocks have an output value range of [0…1], so scaling to the actual module voltage output range is necessary:

`Voltage value [V] = ADC_VALUE x 5.0`

These are the heat current feedback signals of the PS-LSD 6/1 module which are to be processed in the real-time model (requires two A/D channels):

- UHEGO_Heat_I_chx [A] (requires a gain of 6.25)

Note

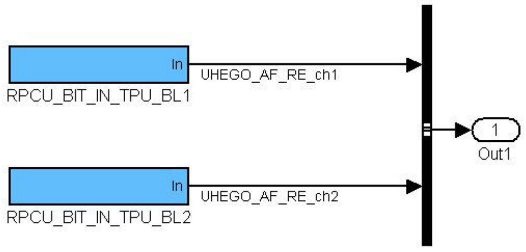
RPCU_ADC blocks have an output value range of [0…1], so scaling to the actual module current output range is necessary:

`Current value [A] = ADC_VALUE x 6.25`

Bit_IO_input subsystem

The path of this subsystem in the demo model is `UHEGO_example_model/Input+Scaling/Bit_IO_input`.

Two RPCU_BIT_IN_TPU_BLx blocks are needed (one for each lambda probe) to measure the two digital signals that are to be processed in the real-time model.



The read mode of the RPCU_BIT_IN_TPU_BLx is set to Read current value (read on request).

This is the digital output signal of the SC-UHEGO 2/1 module which is to be processed in the real-time model (requires 1 channel):

- UHEGO_AF_RE_chx

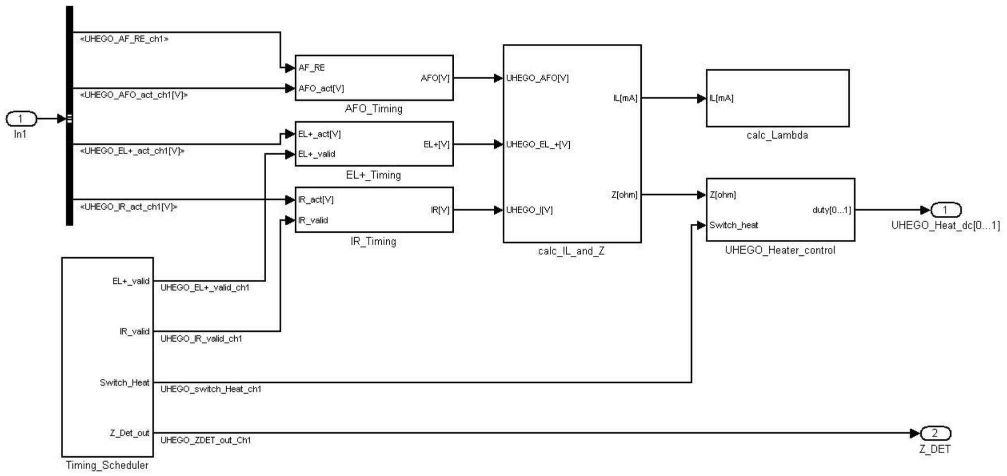
UHEGO Demo Application: Processing the Measurement Results

Objective

This section explains how the measurement restrictions (refer to Measurement restrictions on page 311) are fulfilled in the UHEGO demo application and describes the relevant model parts.

Timing Scheduler subsystem

The Timing_Scheduler subsystem (bottom left in the screenshot below) is crucial for processing the measurement results. The path of this subsystem in the demo model is `UHEGO_example_model/UHEGO_Chx/Timing_Scheduler`.

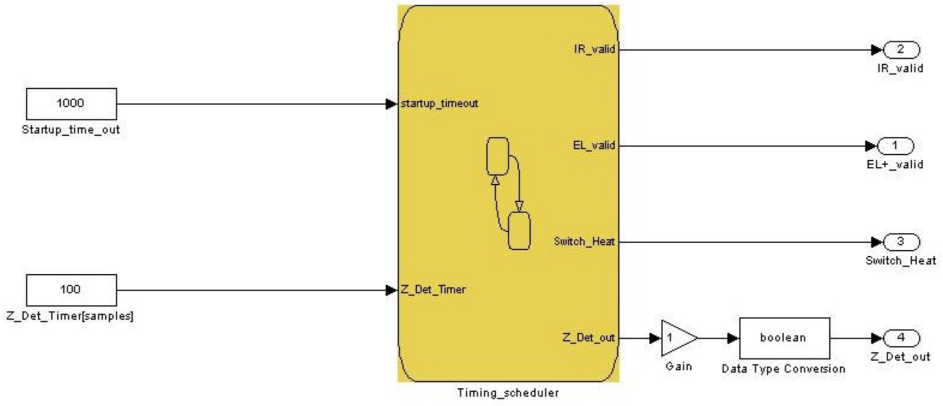


Trigger and status signals

To fulfill the measurement restrictions, a trigger signal and several status signals are used. All these signals leave the Timing_Scheduler subsystem as binary output signals (see graphic above).

Signal	... Is Generated By	... Is Used To
Z_DET_out	Real-time model	Triggers the measurement of the internal resistance (UHEGO_I). The trigger event is a falling edge.
AF_RE	SC-UHEGO 2/1 module	Indicates whether the result of the λ measurement (UHEGO_AFO) is valid (1 means valid).
EL+_valid	Real-time model	Indicates whether the result of the λ measurement (UHEGO_EL+) is valid (1 means valid).
IR_valid	Real-time model	Indicates whether the result of the internal resistance measurement (UHEGO_I) is valid (1 means valid).
Switch_Heat	Real-time model	Indicates whether the heater control signal (PWM signal) is stopped (1 means stopped).

Generation of trigger and status signals is performed in the Timing_scheduler Stateflow chart (see graphic below). The chart incorporates a timer whose temporal step width is 1 ms (base sample time of the model). When the model is executed, it first passes through a start-up phase (Length: 1000 steps). Then normal operation begins, characterized by periodically triggered measurement cycles (Period: 100 steps).

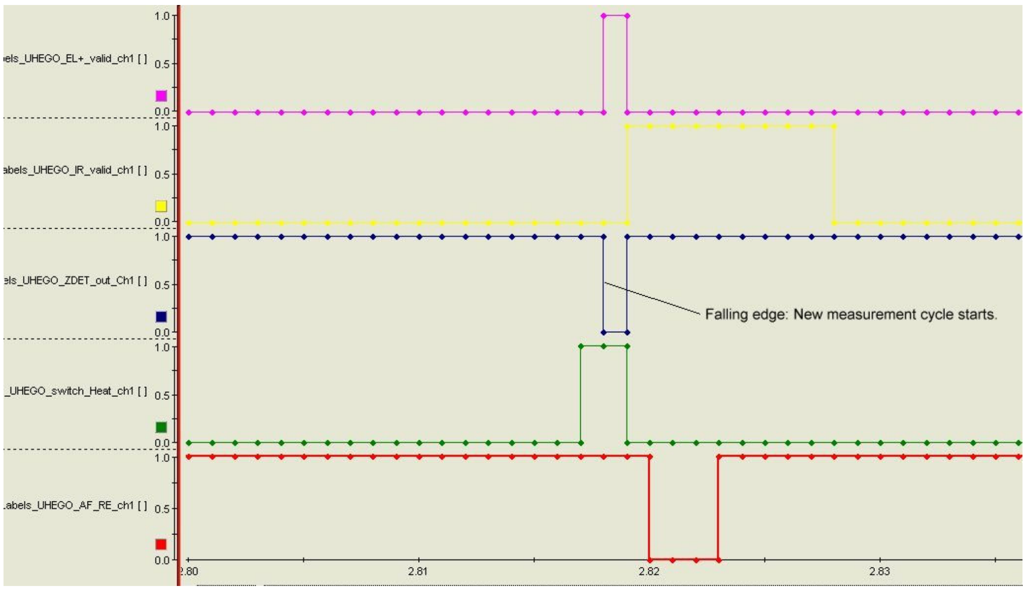


Timing during start-up The start-up phase lasts 1000 steps (1000 ms). During this time the output signals of the Timing_scheduler chart are set as follows:

Signal	Value
Z_DET_out	1 (Trigger event: Falling edge 1→0)
EL+_valid	0 (result invalid)
IR_valid	0 (result invalid)
Switch_Heat	0 (PWM allowed)

During start-up no trigger event occurs and hence no measurements are performed. To guarantee a high level for Z_DET_out right from the beginning of a real-time simulation, the Initial output value parameter of the RPCU_BIT_OUT_TPU_BL1 block is set to High.

Timing during normal operation The following diagram shows the output signals of the Timing_scheduler chart during normal operation and the AF_RE signal generated by the SC-UHEGO 2/1 module. The execution steps of the model are represented by dots in the curves. One measurement cycle comprises 100 steps (100 ms).

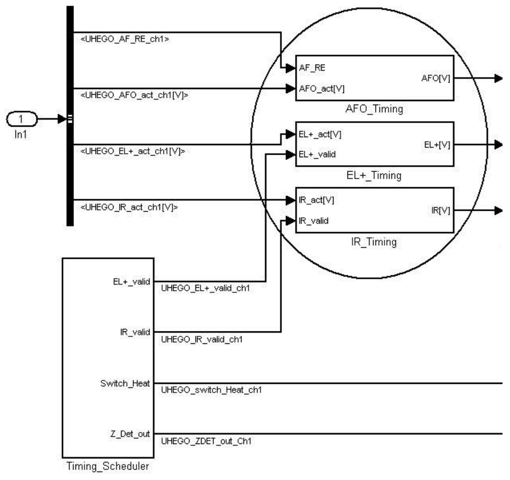


The UHEGO_AF_RE and UHEGO_IR_valid signals take the characteristics of the SC-UHEGO 2/1 module into account.

To be sure that the heat current is constant during the measurement of UHEGO_I, the PWM signal is stopped (UHEGO_switch_Heat=1) one step before the trigger event (falling edge of UHEGO_ZDET).

Data validity analysis

The status signals, e.g., UHEGO_EL+_valid, determine whether the current measurement result is used, or the last valid one which is stored in a buffer. The graphic below shows the relevant blocks in the Simulink demo model.



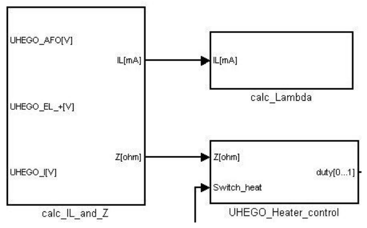
EL+_Timing subsystem The EL+_Timing subsystem (`UHEGO_example_model/UHEGO_Chx/EL+_Timing subsystem`) outputs either the currently measured result or the last valid result, depending on the incoming UHEGO_EL+_valid_ch1 status signal. You can bypass this mechanism and always use the currently measured result by setting the use_current_value_switch parameter to 1.

AFO_Timing subsystem The AFO_Timing subsystem (`UHEGO_example_model/UHEGO_Chx/AFO_Timing subsystem`) outputs either the currently measured result or the last valid result, depending on the incoming UHEGO_AF_RE_ch1 status signal. You can bypass this mechanism and always use the currently measured result by setting the use_current_value_switch parameter to 1.

IR_Timing subsystem The IR_Timing subsystem (`UHEGO_example_model/UHEGO_Chx/IR_Timing subsystem`) outputs either the currently measured result or the last valid result, depending on the incoming UHEGO_IR_valid_ch1 status signal. You can bypass this mechanism and always use the currently measured result by setting the use_current_value_switch parameter to 1.

Calculation of λ

First, the pump current IL is calculated as a function of the UHEGO_AFO[V] and UHEGO_EL_+[V] measurement signals in the calc_IL_and_Z subsystem. Then λ is calculated as a function of the pump current IL in the calc_Lambda subsystem by using a probe-specific lookup table. For the required table data, refer to the technical documentation of the lambda probes. The graphic below shows the relevant blocks in the Simulink demo model.



Calculation of the internal resistance

The graphic above shows the relevant blocks in the Simulink demo model. The internal resistance is calculated as a function of the UHEGO_I[V] measurement signal in the calc_IL_and_Z subsystem. The result is used to calculate the UHEGO_Heat_dc[0...1] heater control signal in the UHEGO_Heater_control subsystem.

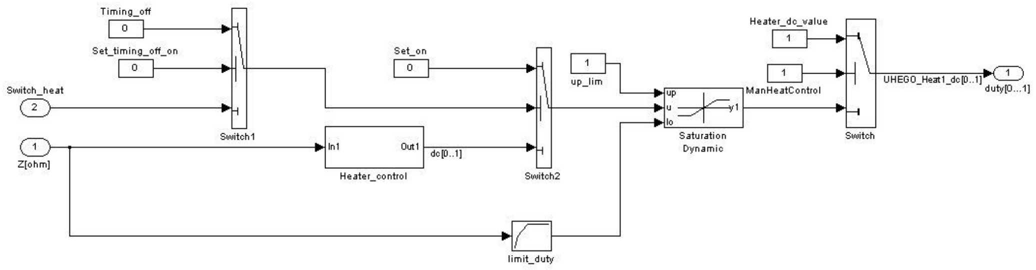
UHEGO Demo Application: Controlling the Heater

Overview

The UHEGO_Heater_control subsystem (`UHEGO_example_model/UHEGO_Ch1/UHEGO_Heater_control`) outputs the UHEGO_Heat1_dc[0..1] signal, which defines a duty cycle. This signal is connected to the RPCU_PWM_TPU_BL1 block (`UHEGO_example_model/Output+Scaling/RPCU_PWM_TPU_BL1`). This block generates the LS_Ctrl PWM signal, which is transferred to the PS-LSD 6/1 power stage module for controlling the heater.

Note

For the PS-LSD 6/1 power stage module, a duty cycle of 0 means that the full current is applied, 1 means no current.



Override options

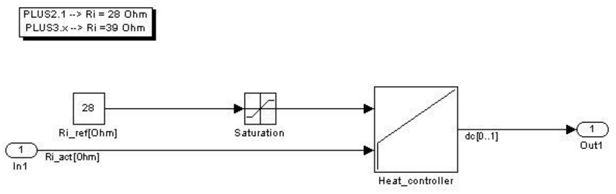
The UHEGO demo application provides the following override options, which are helpful for application testing and validation purposes.

Timing override If the Set_timing_off_on parameter (control input for the Switch1 block) in the UHEGO_Heater_control subsystem is set to 0, the Switch_heat status signal is used as output by the Timing_scheduler Stateflow chart. If the parameter is set to 1, the Timing_off parameter is used as the time-invariant status signal, instead (0 means that the PWM signal is always allowed).

Duty cycle override If the ManHeatControl parameter (control input for the Switch block) in the UHEGO_Heater_control subsystem is set to 0, the duty cycle is used as calculated (see below). If the parameter is set to 1, the Heater_dc_value is applied as the duty cycle (0 means full heating, 1 no heating).

Calculation of the duty cycle

The Heater_control subsystem contains the Heater_controller PID controller, which outputs a duty cycle as a function of the measured internal resistance of the lambda probe (Ri_act[Ohm]) and the probe's internal resistance at working point (700 °C = 1292 °F, Ri_ref[Ohm]).



Duty cycle during warm-up To avoid a thermal shock when the heating process is starting and the temperature is very low (e.g., -40 °C = -40 °F), the duty cycle is limited by the SaturationDynamic block in the UHEGO_Heater_control subsystem so that the heat current is limited, too. Note that the lower duty cycle limit defines the upper current limit. The lower duty cycle limit is calculated by the limit_duty block (look-up table) as a function of the measured resistance value Z[ohm] (R_{ac}).

Duty cycle during resistance measurement While the internal resistance of the lambda probe is measured, the duty cycle of the heater control signal must be set to either 0 or 1 (constant). This value is defined by the Set_on parameter (0 means full heating, 1 no heating) in the UHEGO_Heater_control subsystem (Switch2 block) of the demo model.

Current feedback measurement

A current feedback measurement is performed. The PS-LSD 6/1 module outputs the LS_U(I) signal, which is fed into the real-time model via the RPCU_ADC_BL2 block. However, this signal is not further processed in the demo application.

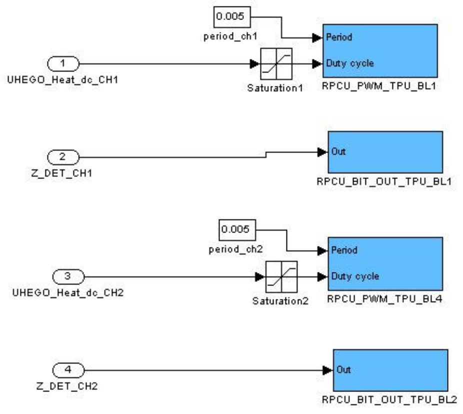
UHEGO Demo Application: Feeding the Control Signals to the Hardware

Objective

This section explains how the LS_Ctrl heater control signal (PWM signal) and the UHEGO_ZDET trigger signal for measuring the internal resistance of the probe are generated. The relevant model parts of the `UHEGO_example_model.mdl` are introduced, as well as crucial implementation aspects.

Output+Scaling subsystem

The path of this subsystem in the demo model is `UHEGO_example_model/Output+Scaling`. This subsystem on the root level of the model feeds the control signals which are associated with the two lambda probes to the RapidPro hardware. The blocks of the RPCU Control Unit Blockset (colored in the graphic below) let you reference the related RapidPro hardware (see the Unit page in the block dialogs).



Heater control signal

The UHEGO_Heat_dc_CHx duty cycle signal is connected to the RPCU_PWM_TPU_BLx block (one block for each lambda probe). The block generates the LS_Ctrl heater control signal (PWM signal), which is routed to the PS-LSD 6/1 PS module.

Measurement trigger signal

The Z_DET_CHx signal is connected to the RPCU_BIT_OUT_TPU_BLx block (one block for each lambda probe). The block generates the UHEGO_ZDET signal, which is routed to the SC-UHEGO 2/1 module. A falling edge of the Z_DET_CHx signal triggers the measurement of the internal resistance of the probe.

Note

To guarantee a high level for UHEGO_ZDET right from the beginning of a real-time simulation, the Initial output value parameter in the dialog of the RPCU_BIT_OUT_TPU_BLx block is set to High.

Measuring Preignitions with the SC-KNOCK 4/1 Module

Where to go from here

Information in this section

Information in other sections

Measuring Preignitions with the SC-KNOCK 4/1 Module (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Gives information about the RTI blocks concerned with the measurement of improper ignition timing (preignition).

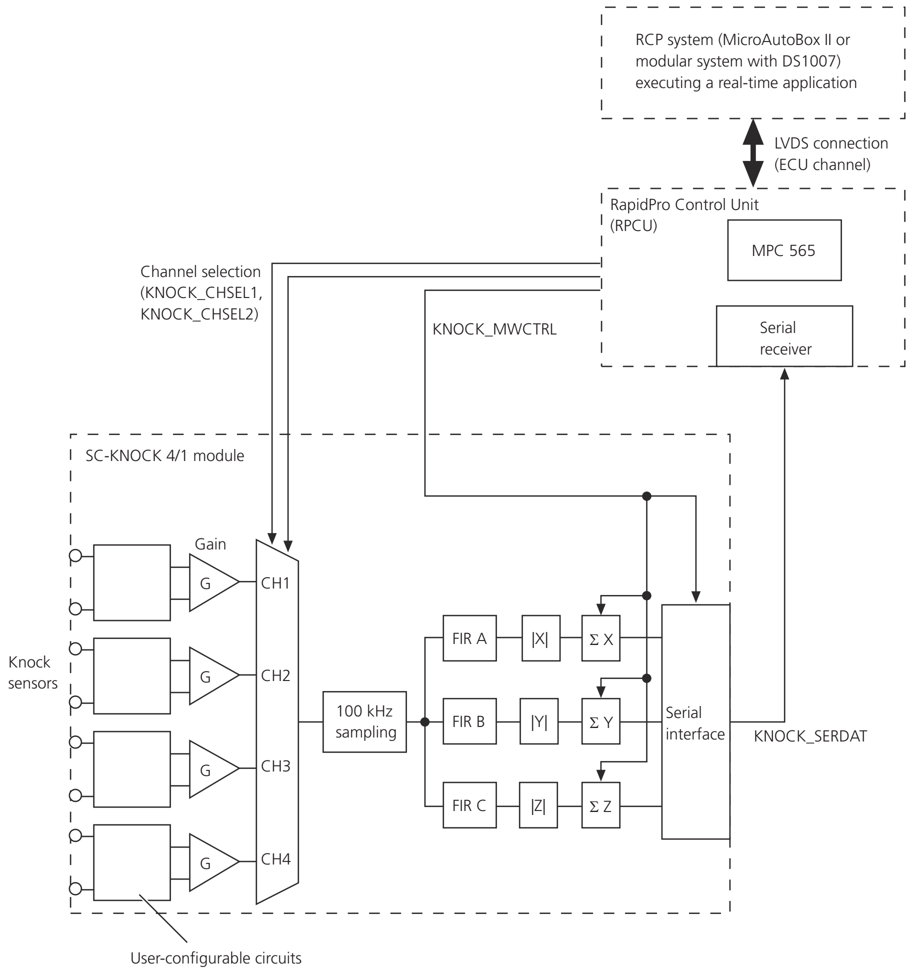
Basics of Knock Signal Measurement

Aim of knock signal measurement	The aim of knock signal measurement is to avoid improper ignition timing (preignition).
Origin of knock signals	Ignition that occurs too early (preignition) is accompanied by a specific engine noise. Knock sensors convert this noise into a voltage signal (knock signal). In fact, knock sensors are acceleration sensors.
SC-KNOCK 4/1 module	Up to four piezoelectric knock sensors (single-ended or differential) can be connected to one SC-KNOCK 4/1 module, which is part of a RapidPro system.

For hardware details on the module, refer to SC-KNOCK 4/1 Module (RapidPro System Hardware Reference 🕮).

Signal processing

The SC-KNOCK 4/1 module samples a knock signal at 100 kHz. The signal can be amplified by adjusting the channel-specific Gain parameter via ConfigurationDesk for RapidPro. For details, refer to Software Configuration in *SC-KNOCK 4/1 Module* of the RapidPro System Hardware Reference 🕮. The knock signal is then fed simultaneously to three independent FIR filters. You can choose in between 10 sets of filter coefficients in ConfigurationDesk for RapidPro. The magnitude of the output of each FIR filter is integrated in one total. As soon as measurement has finished, the totals of the three FIR filters are written as one data set (KNOCK_SERDAT) to a specific serial receiver on the RapidPro Control Unit, and the receiver generates an interrupt.


RCP system (MicroAutoBox II or modular system with DS1007) executing a real-time application
LVDS connection (ECU channel)
RapidPro Control Unit (RPCU)
MPC 565
Serial receiver
Channel selection (KNOCK_CHSEL1, KNOCK_CHSEL2)
KNOCK_MWCTRL
SC-KNOCK 4/1 module
Gain
G
CH1
CH2
CH3
CH4
Knock sensors
100 kHz sampling
FIR A
FIR B
FIR C
|X|
|Y|
|Z|
Σ X
Σ Y
Σ Z
Serial interface
KNOCK_SERDAT
User-configurable circuits

Note

Each SC-KNOCK 4/1 module requires its own serial receiver on the RapidPro Control Unit. Since the RapidPro Control Unit has four serial receivers, you can use four SC-KNOCK 4/1 modules at maximum.

Routing

A SC-KNOCK 4/1 module requires the following routing:

Channel	Signal	Description
One serial input channel	KNOCK_SERDAT	Used for reception of the measurement result (refer to Getting the measurement results on page 328)
One TPU output channel	KNOCK_MWCTRL	Trigger signal (starts and stops the measurement)
Two digital output channels	KNOCK_CHSEL1 KNOCK_CHSEL2	Selection of the knock sensor (CH1 ... CH2)

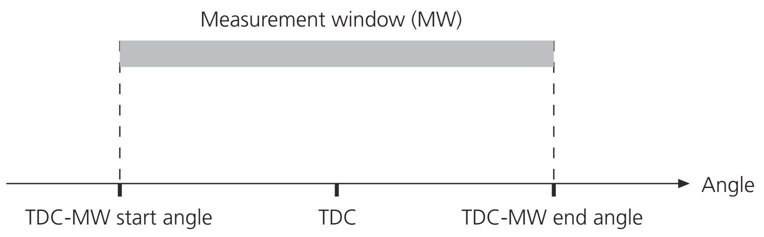
Specification of a knock signal measurement

The following parameters are crucial to the specification of knock signal measurement. They are accessible during run time via the RPCU_SCKNOCK41_CTRL_BLx block. Their initialization and termination values can be specified via the RPCU_SCKNOCK41_SETUP_BLx block:

- Channel selection (sensor)
- TDC
- MW start angle
- MW end angle

The resolution of TDC, MW start angle, and MW end angle is 0.1°.

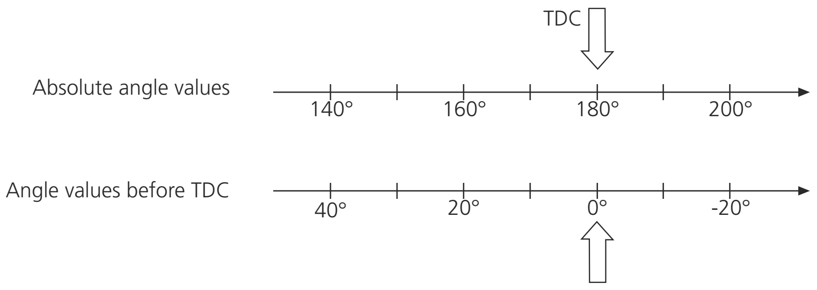
The measurement of a knock signal starts at a specific angle value (MW start angle) and ends at a specific angle value (MW end angle). The range between the start and end angle values is called the measurement window (MW).



TDC = 0° ... 719.9°
MW start angle = –360° ... 359.9°
MW end angle = –360° ... 359.9°

TDC as reference

All angle values that are used to specify measurement windows are relative to “before TDC”. You get these values by subtracting the desired absolute angle value from the absolute TDC angle value.



Example

Suppose a measurement window must start at 140° and end at 230°, and TDC is to be set to 180°. To achieve this, you have to specify +40° as the MW start angle and -50° as the MW end angle:

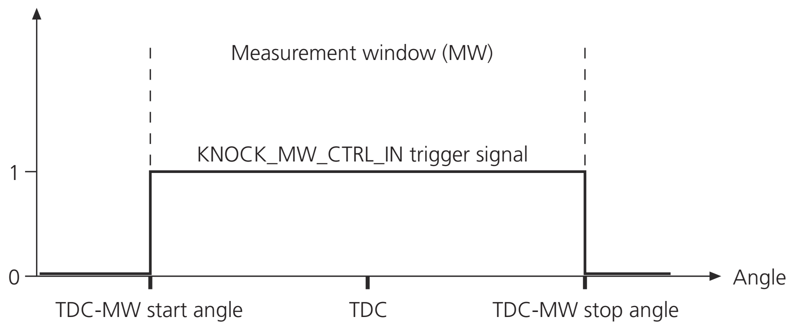
- MW start angle: +40° = 180° - 140°
- MW end angle: -50° = 180° - 230°

Note

The TDC specified by the RPCU_SCKNOCK41_CTRL_BLx block does not interfere with TDCs specified by RPCU INJ_IGN_TPU_BLx blocks.

Trigger signal required by the SC-KNOCK 4/1

The SC-KNOCK 4/1 module requires a trigger signal (measurement window signal) for starting and stopping the measurement. The SC-KNOCK 4/1 module evaluates the trigger signal KNOCK_MWCTRL. If the trigger signal switches from 0 to 1 at the MW start angle, the SC-KNOCK 4/1 module starts knock signal measurement. If it switches from 1 to 0 at the MW end angle, the SC-KNOCK 4/1 module stops knock signal measurement, and the result is written to the serial receiver.



Getting the measurement results

When measurement stops, the result (KNOCK_SERDAT) is written to the buffer of a serial receiver on the RapidPro Control Unit specified in the dialog of the RPCU_SCKNOCK41_SETUP_BLx block, and a serial data interrupt (SC-KNOCK 4/1 SERDAT) is generated. You can read the buffer of the serial receiver by implementing the RPCU_SCKNOCK41_DATA_BLx block in the model.

Related topics

HowTos

How to Implement Knock Signal Measurement.................330
How to Implement the Reading of the Measurement Results.................333

Examples

Example of Implementing Knock Signal Measurement.................334

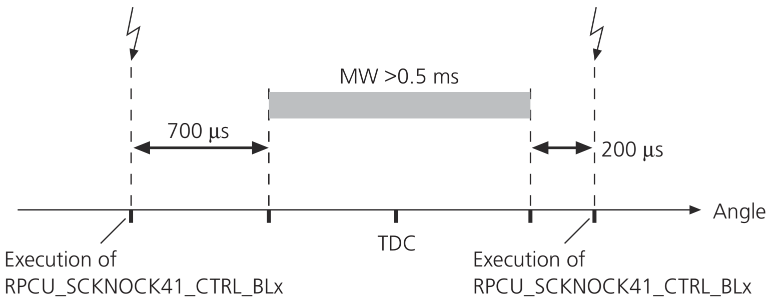
Angle-Triggered Knock Signal Measurement

Efficient implementation

For efficiency reasons, the RPCU_SCKNOCK41_CTRL_BLx block should be updated with respect to the measurement window MW. Thus, it must not reside on the topmost level in your Simulink model, but in an angle-triggered subsystem.

Restrictions

To ensure proper knock signal measurement, the following restrictions apply:


MW >0.5 ms
700 µs
200 µs
Angle
TDC
Execution of RPCU_SCKNOCK41_CTRL_BLx
Execution of RPCU_SCKNOCK41_CTRL_BLx

- The time between the execution of a RPCU_SCKNOCK41_CTRL block and the start of its measurement window must be at least 700 µs.
- The width of a measurement window must be at least 0.5 ms. For example, if the maximum engine speed is 10000 rpm (= 60°/ms), MW start angle and MW end angle must cover a range of at least 30°.
- Measurement windows of RPCU_SCKNOCK41_CTRL blocks referencing the same SC-KNOCK 4/1 module must not overlap. A measurement window must end 200 µs before the next block execution occurs (same or another RPCU_SCKNOCK41_CTRL block) at the latest. This is independent of the knock sensor chosen (channel selection during run time).

- If you use multiple RPCU_SCKNOCK41_CTRL blocks referencing the same SC-KNOCK 4/1 module, each block must be in its own angle-triggered subsystem.

  Note

  Multiple RPCU_SCKNOCK41_CTRL blocks can be in the same subsystem if they reference different SC-KNOCK 4/1 modules.

- A new measurement window MW must not start until an RPCU_SCKNOCK41_DATA_BLx block has been executed and the results of the preceding measurement window MW have been read. Otherwise, an error occurs (buffer overflow).
- The simState variable must not be set to 1 (PAUSE) when the model is running. Otherwise, an error occurs (buffer overflow).

Event-Triggered Reading of the Measurement Results

Objective

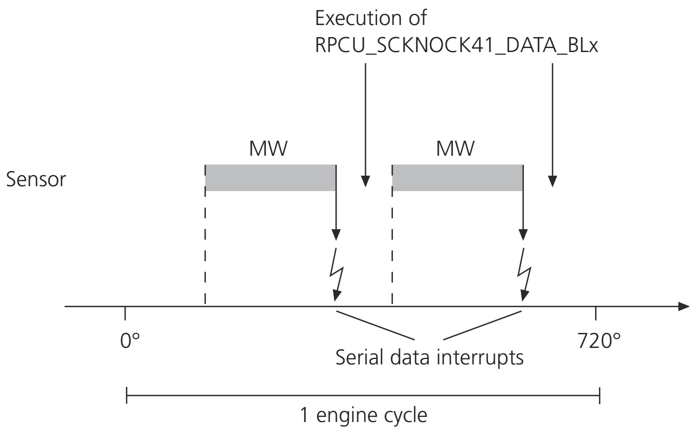
To ensure efficient reading of the measurement results, an RPCU_SCKNOCK41_DATA_BLx block should not be executed until a measurement result is actually available.

Serial data interrupt

A serial data interrupt (SC-KNOCK 4/1 SERDAT) can be generated each time measurement stops and the measurement result (KNOCK_SERDAT) is written to the buffer of the specified serial receiver on the RapidPro Control Unit.

Example

Suppose knock signal measurement is performed twice during one engine cycle, i.e. two measurement windows MW are specified. In this case a serial data interrupt is generated after each of the two measurement windows MW.



Efficient implementation

You can use the serial data interrupt to trigger the reading of the buffer. Thus, an RPCU_SCKNOCK41_DATA_BLx block should reside in a Simulink function-call subsystem triggered by a serial interrupt.

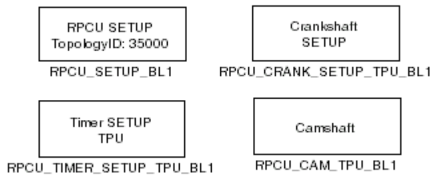
How to Implement Knock Signal Measurement

Objective

You can measure a knock signal at a specific engine angle for a specific angle range.

Preconditions

You must have implemented engine control.



For details, refer to How to Set Up Camshaft Signal Processing on page 207.

Restriction

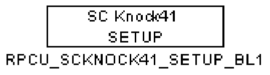
Note

Up to 4 RPCU_SCKNOCK41_SETUP Block blocks and up to 12 RPCU_AABP_TPU_BLx blocks are allowed in one model, but they must not exceed an overall total of 12.

Method

To implement knock signal measurement

1 Insert a RPCU_SCKNOCK41_SETUP block, double-click it to open its dialog, and specify the parameters.



2 In the Board/Module number list, choose the desired board/module number.

3 In the ECU channel list, choose the desired ECU channel number.

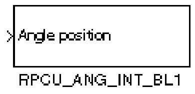
(Only if you work with a DS1007 modular system)

4 In the Module selection list, choose the SC-KNOCK 4/1 module. The module is marked by "+". Modules that are used otherwise are marked by "*", for example, a channel routed to a SC-KNOCK 4/1 module is used by another block.

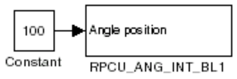
5 Switch to the Parameters page and specify the measurement settings for initialization and termination.

6 Switch to the Advanced page and select the Enable interrupt checkbox.

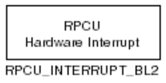
7 Insert a RPCU_ANG_INT block, double-click it, and specify its parameters.



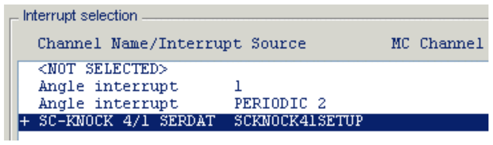
8 Insert a Simulink Constant block, specify the trigger angle, and connect the block to the RPCU_ANG_INT block.



9 Insert an RPCU_INTERRUPT block.

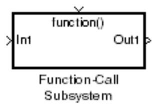


10 Double-click it, and select the interrupt that is associated with the RPCU_ANG_INT block.


Interrupt selection

```
Channel Name/Interrupt Source          MC Channel
<NOT SELECTED>
Angle interrupt       1
Angle interrupt       PERIODIC 2
+ SC-KNOCK 4/1 SERDAT  SCKNOCK41SETUP
```

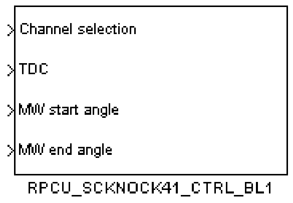
11 Insert a Simulink Function-Call Subsystem block.



12 Connect the RPCU_INTERRUPT block to the trigger inport of the Function-Call Subsystem block.

13 Double-click the Function-Call Subsystem block to open it.

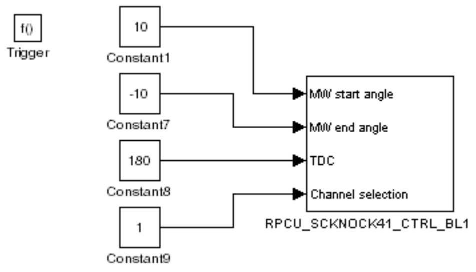
14 Insert a RPCU_SCKNOCK41_CTRL block in the triggered subsystem, double-click it to open its dialog, and specify the parameters.



Note

Only modules that are occupied by an RPCU_SCKNOCK41_SETUP block are available in the Module selection list on the Unit page.

15 Insert four Simulink **Constant** blocks in the triggered subsystem, specify the channel (Ch1 … Ch4), the TDC, the MW start angle, and the MW end angle, and connect the Constant blocks to the **RPCU_SCKNOCK41_CTRL** block.



Note

- The time between the execution of a RPCU_SCKNOCK41_CTRL block and the start of its measurement window must be at least 700 µs.
- The width of a measurement window must be at least 0.5 ms.
- A measurement window must end 200 µs before the next block execution occurs (same or another RPCU_SCKNOCK41_CTRL block) at the latest.

For details, refer to Restrictions on page 328.

16 Close the triggered subsystem and change to the topmost level of the Simulink model.

Result

You have implemented knock signal measurement for one knock sensor. Its knock signal is measured once during an engine cycle.

Next step

After you have implemented knock signal measurement, you need to implement reading the measurement results from the buffer of the serial receiver on the RapidPro Control Unit.

Related topics

HowTos

Examples

References

RPCU_SCKNOCK41_CTRL Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_SCKNOCK41_DATA Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_SCKNOCK41_SETUP Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

How to Implement the Reading of the Measurement Results

Objective

You can read the result of a measurement window from the buffer of the serial receiver on the RapidPro Control Unit.

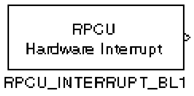
Preconditions

You have already implemented knock signal measurement, refer to How to Implement Knock Signal Measurement on page 330.

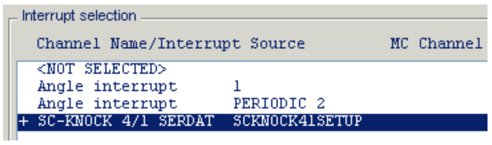
Method

To implement the reading of the measurement results

1 Insert an RPCU_INTERRUPT block.

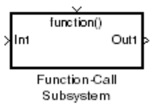


2 Double-click it, and select the SC-KNOCK 4/1 SERDAT interrupt which is associated with the RPCU_SCKNOCK41_SETUP block.


Interrupt selection

```
 Channel Name/Interrupt Source         MC Channel
 <NOT SELECTED>
 Angle interrupt      1
 Angle interrupt      PERIODIC 2
+ SC-KNOCK 4/1 SERDAT  SCKNOCK41SETUP
```

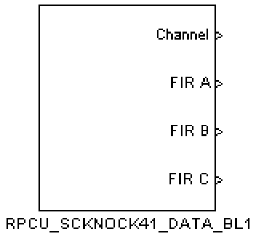
3 Insert a Simulink Function-Call Subsystem block.



4 Connect the RPCU_INTERRUPT block to the trigger inport of the Function-Call Subsystem block.

5 Double-click the Function-Call Subsystem block to open it.

6 Insert a RPCU_SCKNOCK41_DATA block in the triggered subsystem, double-click it to open its dialog, and specify the parameters.



Note

Only modules that are referenced by an RPCU_SCKNOCK41_CTRL block are available in the Module selection list on the Unit page.

Result

You have implemented the reading of the measurement results, which can be read from the outports of the RPCU_SCKNOCK41_DATA block.

Related topics

HowTos

How to Implement Knock Signal Measurement....................330

Examples

Example of Implementing Knock Signal Measurement....................334

References

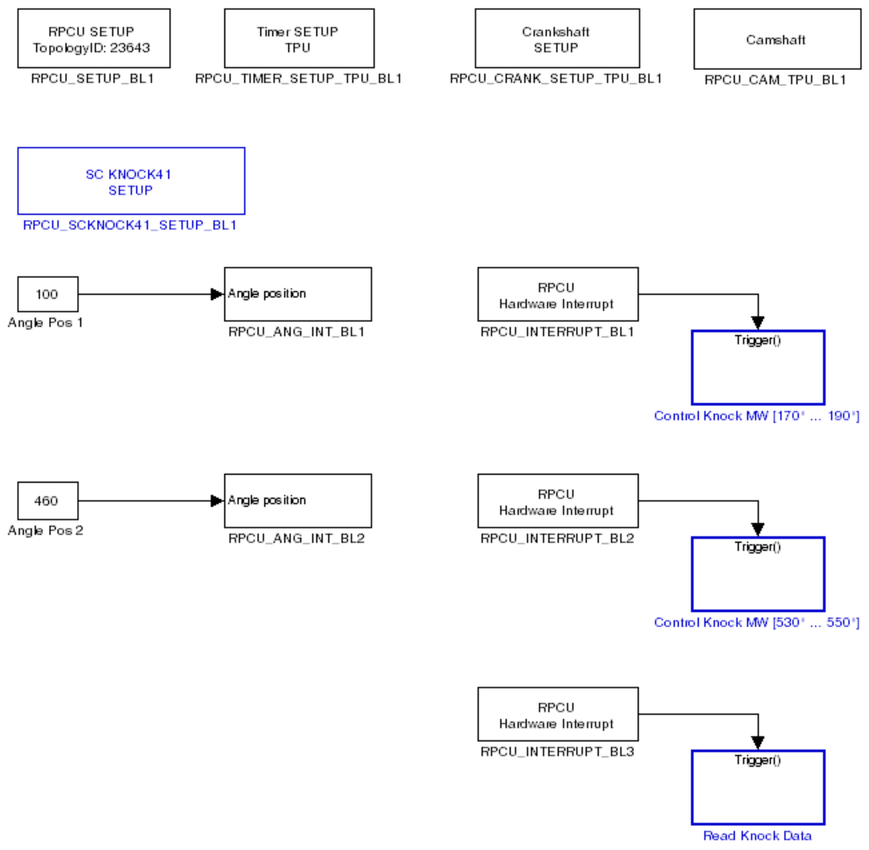
RPCU_SCKNOCK41_CTRL Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
RPCU_SCKNOCK41_DATA Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
RPCU_SCKNOCK41_SETUP Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Example of Implementing Knock Signal Measurement

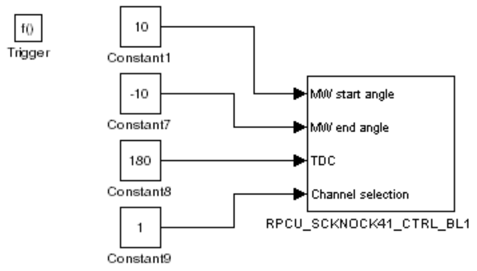
Objective

Suppose you have one knock sensor connected to channel Ch1, and you want to measure the knock signal twice during an engine cycle, at [170° … 190°] and [530° … 550°].

The topmost level of the Simulink model looks something like this:



The first angle-triggered subsystem (**Control Knock MW [170° … 190°]**), see below, is implemented as follows: MW start angle = 10°, and MW end angle = -10°, TDC = 180°, and channel Ch1. It is triggered at 100° (**Angle Pos 1** block).



The second angle-triggered subsystem (**Control Knock MW [530° … 550°]**) is implemented analogously: MW start angle = 10°, and MW end angle = -10°, TDC = 540°, and channel Ch1. It is triggered at 460° (**Angle Pos 2** block).

The event-triggered subsystem (**Read Knock Data**) contains the RPCU_SCKNOCK41_DATA block. The serial data interrupt of the serial receiver on the RapidPro Control Unit acts as the trigger source.

Related topics

HowTos

References

RPCU_SCKNOCK41_CTRL Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_SCKNOCK41_DATA Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_SCKNOCK41_SETUP Block (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Measuring Temperatures with the SC-TC 8/1 Module

Objective

You can measure temperatures in run time by using up to 8 thermocouples connected to the SC-TC 8/1 module.

Where to go from here

Information in this section

Basics on Temperature Measurement with the SC-TC 8/1 Module

Hardware features

Temperature measurement with the SC-TC 8/1 module is based on the following hardware features:

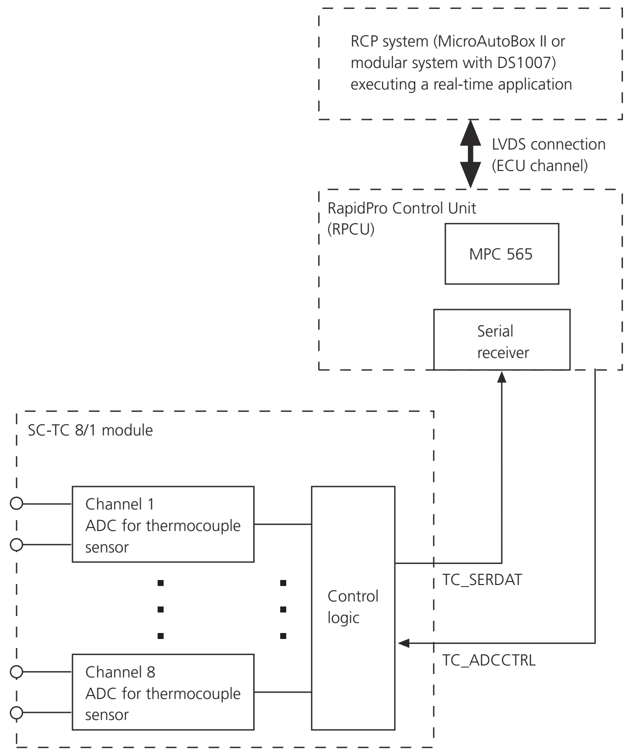
- Temperature range: -100 … +1372 °C (equivalent to -148 … +2502 °F – thermocouple of type K: NiCr-Ni)
- 8 galvanically isolated input channels for 8 thermocouples
- One 24-bit A/D converter per channel (resolution: 0.1 °C, equivalent to 0.18 °F), sampling rate for each converter is 0.1/7.6 ... 60 Hz (software-configurable at initialization time, the lower limit depends on the conversion mode – see below). Resolution is 0.1 Hz.
- Trigger input for evoking measurement and result transfer, the master application (RCP system) sends the trigger signal

For details on the SC-TC 8/1 module hardware, refer to SC-TC 8/1 Module (RapidPro System Hardware Reference 🕮).

Signal processing

The analog voltage signals of the thermocouples are digitized (ADC). As soon as a measurement has finished, the results of the channels used are written as one data set (TC_SERDAT) to the serial receiver on the RapidPro Control Unit, and the receiver generates an interrupt.

The master application running on the RCP system can send a trigger signal (TC_ADCCTRL) to the SC-TC 8/1 module to start measurement or result transfer (see "Configuring of the measurement" below).


RCP system (MicroAutoBox II or modular system with DS1007) executing a real-time application
LVDS connection (ECU channel)
RapidPro Control Unit (RPCU)
MPC 565
Serial receiver
SC-TC 8/1 module
Channel 1 ADC for thermocouple sensor
Channel 8 ADC for thermocouple sensor
Control logic
TC_SERDAT
TC_ADCCTRL

Software features (RTI support)

The RapidPro Control Unit Blockset (RTI RPCU Blockset) contains the RPCU_SCTC81_Data_BLx block, refer to RapidPro System – I/O Subsystem MPC565 RTI Reference. This block can be used in applications running on MicroAutoBox II or DS1007 (RCP systems) and allows you:

- To configure temperature measurement with the SC-TC 8/1 module
- To access the measurement result (vector output comprises results of all channels used)
- To monitor the measurement status (vector output comprises states of all channels used)
- To monitor the status of the dual-port memory (DPMEM), which is used for master-slave communication, for example, result transfer from the RapidPro system to the RCP system

You can use multiple RPCU_SCTC81_DATA_BLx blocks in one Simulink model that reference the same SC-TC 8/1 module. The blocks must have identical dialog settings.

Configuring the measurement

Measurement with the SC-TC 8/1 module is configured during initialization of the master application, based on the dialog settings of the RPCU_SCTC81_DATA_BLx block.

Note

The configuration of the SC-TC 8/1 module cannot be modified during run time.
Configuration of the SC-TC 8/1 module via ConfigurationDesk for RapidPro is not possible.

You can configure the temperature measurement with the SC-TC 8/1 module as follows:

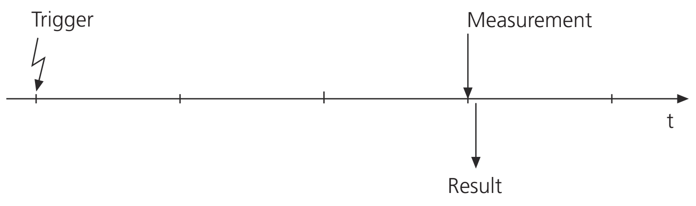
- Triggered conversion mode with auto transfer

  After the SC-TC 8/1 module is triggered by the master application, one measurement starts with a delay of 3 sampling steps (3 / conversion rate). The result is automatically returned.

  Note

  In this conversion mode, the conversion rate is limited to 7.6 ... 60 Hz. If a second trigger occurs during the delay time, it is ignored.

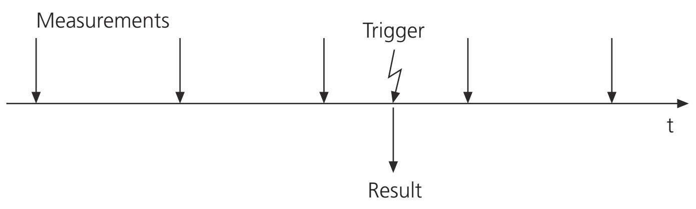
  Triggered conversion mode with auto transfer

- Continuous conversion mode with triggered transfer

  Measurements are performed continuously (conversion rate: 0.1 ... 60 Hz). After the SC-TC 8/1 module is triggered by the master application, the result of the most recently finished measurement is returned.

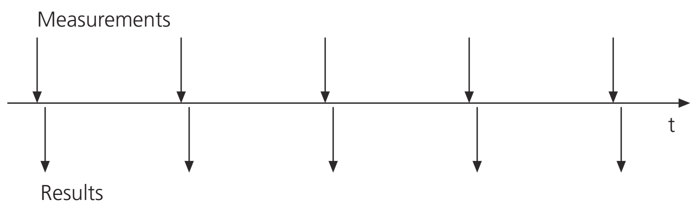
  Continuous conversion mode with triggered transfer

- Continuous conversion mode with auto transfer

  Measurements are performed continuously (conversion rate: 0.1 ... 60 Hz). After a measurement is finished, the result is automatically returned.

Continuous conversion mode with auto transfer



Note

Measurement starts synchronously for all channels used. A measurement result is a vector comprising the single results of all these channels.

Measurement delay due to calibration

Whenever a real-time application is started on the RCP system, all A/D converters on the SC-TC 8/1 module are calibrated, i.e., internal offsets are compensated. While the A/D converters are being calibrated, results are not returned. This delay depends on the **Conversion rate** parameter specified in the dialog of the RPCU_SCTC81_Data_BLx block and is maximum (about 2 seconds) for the lowest conversion rate.

Trigger signal requirements

If the module is used in "Continuous conversion mode with triggered transfer" or "Triggered conversion mode with auto transfer", a digital output channel (**SC-TC 8/1 ADCCTRL**) must be routed to the SC-TC 8/1 module, refer to RapidPro System Information Required for Implementation with RTI and RTLib on page 21.

The trigger signal must be generated as digital OUT by an appropriate block of the RPCU blockset, for example, the RPCU_PWM_TPU_BLx block. You can specify the edge polarity of the trigger signal via the RPCU_SCTC81_Data_BLx block.

Result interrupt

A serial data interrupt (SC-TC 8/1 SERDAT) can be generated each time new measurement data is available in the DPMEM. For information on interrupt handling, see Interrupts Provided by the RapidPro Control Unit on page 349.

Triggered result transfer

Whenever an RPCU_SCTC81_DATA block is executed, it reads the result that is currently available in the DPMEM.

Tip

To have access to the most recent measurement result, the RPCU_SCTC81_DATA block must reside in a Simulink function-call subsystem:
- The RPCU_SCTC81_DATA block must enable SC-TC 8/1 SERDAT interrupts (**Enable interrupt** checkbox is selected in its block dialog)
- The Simulink function-call subsystem must be triggered by the SC-TC 8/1 SERDAT interrupts

I/O mapping

The I/O mapping between the specified signals and the I/O connector pins depends on the installed SC and PS modules and the connected RapidPro units. The I/O mapping information is specific to your RapidPro system and can be displayed using ConfigurationDesk for RapidPro. For more information, refer to Signal Mapping to I/O Pins on page 20.

Related topics

Basics

Interrupts Provided by the RapidPro Control Unit.......................................349
RapidPro System Information Required for Implementation with RTI and RTLib.......................21

References

RPCU_SCTC81_Data_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
SC-TC 8/1 Module (RapidPro System Hardware Reference 📖)

How to Implement Temperature Measurement with the SC-TC 8/1 Module

Objective

To return the most recent measurement result, an RPCU_SCTC81_Data_BLx block must reside in a task triggered by its own SC-TC 8/1 SERDAT interrupts.

Preconditions

An RPCU_SETUP_BLx must reside in the Simulink model, refer to How to Import the Hardware Topology to a Model on page 25.

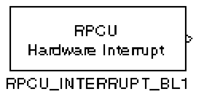
If you want to trigger the SC-TC 8/1 module, the following preconditions must be fulfilled:

1. A digital output channel (SC-TC 8/1 ADCCTRL) must be routed to the SC-TC 8/1 module, refer to RapidPro System Information Required for Implementation with RTI and RTLib on page 21.
2. An appropriate RPCU block must have selected the SC-TC 8/1 ADCCTRL channel as the output driver, for example, an RPCU_PWM_TPU_BLx block.

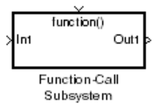
Method

To implement temperature measurement with the SC-TC 8/1 module

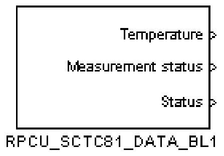
1 Insert an RPCU_INTERRUPT_BLx block.



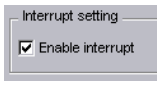
2 Insert a Simulink Function-Call Subsystem block and open it.



3 Insert an RPCU_SCTC81_Data_BLx block in the Function-Call Subsystem and open the block's dialog.



4 Specify the parameters in the dialog of the RPCU_SCTC81_Data_BLx block. On the Parameters page, select the Enable interrupt checkbox.



Interrupt setting

☑ Enable interrupt

Note

Though the Conversion rate resolution is limited to 0.1 Hz, you can enter more fractional digits. However, the RapidPro system does not make use of the unnecessary fractional digits but uses rounded numbers, for example: 2.25 → 2.3, 3.34615 → 3.3.

For implementation reasons, the actual conversion rate slightly differs from the value entered:

- Conversion rate < 55 Hz: Deviation ≤ ±0.05 Hz
- Conversion rate ≥ 55 Hz: Deviation ≤ ±0.064 Hz

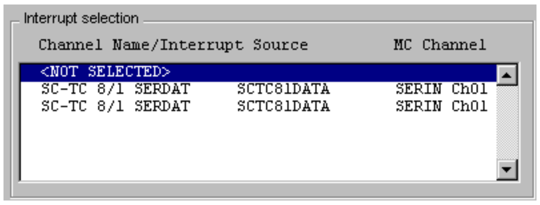
5 Connect the RPCU_INTERRUPT block to the trigger inport of the Function-Call Subsystem block.

6 Open the RPCU_INTERRUPT block.

7 From the **Interrupt selection** list, select the **SC-TC 8/1 SERDAT** interrupt which is associated with the **RPCU_SCTC81_DATA** block.

Note

If multiple RPCU_SCTC81_DATA blocks that reference the same SC-TC 8/1 module are implemented, the same SC-TC 8/1 SERDAT interrupt source is displayed multiple times. It does not matter which one you select:



Interrupt selection

```
Channel Name/Interrupt Source          MC Channel
<NOT SELECTED>
SC-TC 8/1 SERDAT      SCTC81DATA       SERIN Ch01
SC-TC 8/1 SERDAT      SCTC81DATA       SERIN Ch01
```

Result

You have implemented temperature measurement with the SC-TC 8/1 module. The most recent measurement results will be returned (event-triggered reading of the results).

Related topics

Basics

RapidPro System Information Required for Implementation with RTI and RTLib........................ 21

HowTos

How to Import the Hardware Topology to a Model.. 25

References

RPCU_INTERRUPT_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_PWM_TPU_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_SCTC81_Data_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
RPCU_SETUP_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)

Diagnostic Information Provided by the RapidPro System

Objective

The diagnostic feature of the RapidPro Control Unit provides access to the alive status of the RapidPro Units within a stack, and diagnostic functions for reading diagnostic data from the installed SC and PS modules.

Where to go from here

Information in this section

Information in other sections

Diagnostic (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Gives information about the RTI blocks concerned with diagnostic features.

Diagnostic (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
Gives information about the RTLib functions concerned with diagnostic features.

Basics on Diagnostics

Objective

The hardware components of your RapidPro system provide specific diagnostic information, which can be accessed by your application. There is specific diagnostic information on the RapidPro Unit boards and more detailed diagnostic information on the installed SC and PS modules.

Unit diagnostic information

You can request the alive status of a RapidPro Unit by using the RTI blockset and RTLib functions. The master can determine an erroneous connection or a malfunctioning of the RapidPro Unit. This feature requires one RPCU_DIAGNOSIS_ALIVE_BLx block in the Simulink model for each RapidPro Unit to be supervised.

In addition, if the RapidPro System detects a malfunction of the master system (MicroAutoBox II), the RapidPro system deactivates it's output drivers (tristate) and remains in this state until the master system's real time application is restarted. This feature requires at least one RPCU_DIAGNOSIS_ALIVE_BLx block in the Simulink model, addressing an arbitrary RapidPro Unit board.

For implementation details of the RPCU_DIAGNOSIS_ALIVE_BLx block, refer to RPCU_DIAGNOSIS_ALIVE_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Module diagnostic information

You can request diagnostic data on installed RapidPro modules by using the RTI blockset and RTLib functions. The master can request module-specific diagnostic information and channel-specific information for the modules. For example, you can request the diagnostic types "Short to GND", "Overvoltage", and "Open Load". The requestable diagnostic types depend on the installed modules.

Note

For preparing a Simulink model, you need *one* RPCU_DIAGNOSIS_BLx block for each installed SC or PS module. Refer to RPCU_DIAGNOSIS_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).

Diagnostic data array

Each module that provides diagnostic information stores it in a diagnostic data array. This array contains both the module-specific data and the channel-specific data.

If you use the RTI Blockset to access diagnostic data, each diagnostic type supported by the specified module adds a corresponding outport to the diagnostic block.

If you use the RTLib functions, you have to initialize the required diagnostic function and request the diagnostic data before you can read the specific information.

Diagnostic data reset

The diagnostic messages can be divided into warnings and errors. They have different reset behavior.

- Diagnostic warnings occur if a fault, such as “Open load”, is detected. The warning is dropped as soon as the cause of the warning is removed and the diagnostic data has been read.
- Diagnostic errors occur if a fault, such as “Short Circuit”, is detected. The module tries to reactivate the channel for a predefined number of times. If the cause of the error is removed while reactivating the channel, the diagnostic data is reset when you read it. If the cause of the error does not disappear, the channel or the entire module is shut down. You have to remove the source of the error and reset the module's diagnostics via ConfigurationDesk for RapidPro or via your application model.

Note

You can reset diagnostic data of RapidPro modules at run time, refer to RPCU_DIAG_MODULE_RESET_BLx. In addition, you can reset the diagnostic data of all modules globally by using RPCU_DIAG_GLOBAL_RESET_BLx.
Keep in mind that you cannot reset diagnostics channel-wise. If you reset the diagnostics for a particular channel, you always reset all the diagnostics of the module to which the channel belongs. For details, refer to How to Reset Shutdown States of Modules (ConfigurationDesk for RapidPro - Guide 📖).

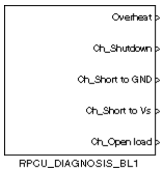
Diagnostic warnings and diagnostic errors

For the available diagnostic warnings and diagnostic errors of the modules, there are two methods:

- Look at the module’s data sheet in the RapidPro System Hardware Reference 📖, for example PS-LSD 6/1 Data Sheet (cut-out).

Parameter	Specification
...	...
Diagnostics	Open load, short to GND, short to supply, overheat, shutdown
...	...

  For example, the PS-LSD 6/1 module supports five diagnostic types: Open load, short to GND, short to supply, overheat, and shutdown.
- Open a Simulink model, drag an RPCU_DIAGNOSIS_BLx block to it and make the corresponding settings for the module you want to read the diagnostic information from. After parameterizing the block, there are outports for each available diagnostic type. Channel-specific diagnostic types start with a “Ch_”.



RTI/RTLib support

You can access the diagnostic feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- Diagnostic (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
- Diagnostic (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)

How to Implement a Run Time Reset Option for Diagnosis Data

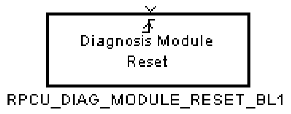
Objective

To reset diagnosis data of RapidPro modules at run time.

Method

To implement a run time reset option for diagnosis data

1 Insert an Parameters Page (RPCU_DIAG_MODULE_RESET_BLx) block.



2 Connect the trigger port of the block with an appropriate signal line. A rising edge signal is required as trigger.

3 Double-click the block.

The block dialog opens, the Unit page is displayed.

4 Specify the RapidPro hardware.

5 Select the Parameters page.

6 Specify the module.

Result

You have implemented a run time reset option for the diagnosis data of a specific module.

Tip

In addition, you can reset the diagnosis data of all modules globally by using the RPCU_DIAG_GLOBAL_RESET_BLx.

Related topics

References

RPCU_DIAG_GLOBAL_RESET_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
RPCU_DIAG_MODULE_RESET_BLx (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)

Interrupts Provided by the RapidPro Control Unit

Objective

This section gives you an overview of interrupts that are provided by the RTI RapidPro Control Unit Blockset and RTLib functions.

Where to go from here

Information in this section

Information in other sections

Interrupt Handling (RapidPro System – I/O Subsystem MPC565 RTI Reference 📖)
Gives information about the RTI block that makes interrupts of the RapidPro system available as trigger sources.

Interrupt Handling (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖)
Gives information about the RTLib functions concerned with the generation of angle-based interrupts.

Basics of Interrupt Handling

Objective

Interrupts from the RapidPro Control Unit (slave) are used to trigger interrupt-driven tasks on the RCP system (master).

Interrupt handling between master and slave

The implemented subinterrupt handling for the ECU interface makes it possible to distinguish between various interrupts which are triggered on the master by the slave.

Interrupt-driven subsystem

Functions which you want to trigger by interrupt must be embedded in an interrupt-driven subsystem in your model. For instructions on doing this, refer to Tasks Driven by Interrupt Blocks (RTI and RTI-MP Implementation Guide 📖). The interrupt which should trigger the subsystem must be made available using the RPCU_INTERRUPT_BLx block. Depending on the RTI blocks used in your model, the RPCU_INTERRUPT_BLx block provides a list of all the interrupts which are enabled in your model.

Tip

When you work with RTLib, an interrupt-driven subsystem corresponds to an interrupt service routine (ISR). Your hand-coded application must contain one basic interrupt service. Within this service you can implement subinterrupts using the appropriate RTLib functions. Each subinterrupt can be serviced separately, for example, by using a Switch Case construct. For further information, refer to Interrupt Handling (RapidPro System – I/O Subsystem MPC565 RTLib Reference 📖).

Overview of available interrupts

The following table gives you an overview of available interrupts:

Feature	Interrupt	Required RTI Block
Bit I/O	Data Ready interrupt Shows that new input values exist if request-read=On.	For enabling: ▪ RPCU_BIT_IN_TPU_BLx For providing: ▪ RPCU_INTERRUPT_BLx
A/D conversion	EOC interrupt Shows that the conversion has finished and the ADC values are available.	For enabling: ▪ RPCU_ADC_BLx For providing: ▪ RPCU_INTERRUPT_BLx
PWM signal generation	PWM interrupt Shows that the specified number of PWM periods has been reached.	For enabling: ▪ RPCU_PWM_TPU_BLx ▪ RPCU_MCPWM_EA_TPU_BLx ▪ RPCU_MCPWM_CA_TPU_BLx For providing: ▪ RPCU_INTERRUPT_BLx
Signal measurement	Data Ready interrupt Shows that new measurement values exist if request-read=On.	For enabling: ▪ RPCU_PWM2D_TPU_BLx ▪ RPCU_PW2D_MIOS_BLx ▪ RPCU_PW2D_TPU_BLx ▪ RPCU_F2D_MIOS_BLx ▪ RPCU_F2D_TPU_BLx For providing: ▪ RPCU_INTERRUPT_BLx
Incremental encoder interface	Index found interrupt Shows that the index signal of the incremental encoder has been used to reset the position counter.	For enabling: ▪ RPCU_ENC_POS_TPU_BLx For providing: ▪ RPCU_INTERRUPT_BLx
Engine control	Angle-based interrupt Shows that the crankshaft has reached one fixed angle position or a periodic angle interval.	For enabling: ▪ RPCU_ANGLE_INT_BLx For providing: ▪ RPCU_INTERRUPT_BLx

Subinterrupt Numbers

Subinterrupt numbers which can be indexed

The following table shows the subinterrupt numbers which can be indexed in master functions:

Define	Value	Comment
DSSINTSMC_TPU_SINT_NO_OFFSET	0-47	Reserved for the TPU channels: ▪ 0-15: TPU A, Channel 1-16 ▪ 16-31: TPU B, Channel 1-16 ▪ 32-47: TPU C, Channel 1-16
DSSINTSMC_QADC_A_SINT_NO	48	QADC A End-of-Conversion interrupt
DSSINTSMC_QADC_B_SINT_NO	49	QADC B End-of-Conversion interrupt
DSSINTSMC_ADDIO_SIO_INT_NO	50-53	Serial IO Receiver interrupt
Reserved	54-55	Reserved
DSSINTSMC_MIOS_FPW2D_INT_NO	56-65	Data Ready interrupt
Reserved	66-95	Reserved
DSSINTSMC_PER_ANGLE_INT_1_NO	96	Periodic angle interrupt 1
DSSINTSMC_PER_ANGLE_INT_2_NO	97	Periodic angle interrupt 2
DSSINTSMC_PER_ANGLE_INT_3_NO	98	Periodic angle interrupt 3
DSSINTSMC_PER_ANGLE_INT_4_NO	99	Periodic angle interrupt 4
DSSINTSMC_PER_ANGLE_INT_5_NO	100	Periodic angle interrupt 5
DSSINTSMC_PER_ANGLE_INT_6_NO	101	Periodic angle interrupt 6
DSSINTSMC_ANGLE_INT_01_NO	112	Angle interrupt 1
DSSINTSMC_ANGLE_INT_02_NO	113	Angle interrupt 2
DSSINTSMC_ANGLE_INT_03_NO	114	Angle interrupt 3
DSSINTSMC_ANGLE_INT_04_NO	115	Angle interrupt 4
DSSINTSMC_ANGLE_INT_05_NO	116	Angle interrupt 5
DSSINTSMC_ANGLE_INT_06_NO	117	Angle interrupt 6
DSSINTSMC_ANGLE_INT_07_NO	118	Angle interrupt 7
DSSINTSMC_ANGLE_INT_08_NO	119	Angle interrupt 8
DSSINTSMC_ANGLE_INT_09_NO	120	Angle interrupt 9
DSSINTSMC_ANGLE_INT_10_NO	121	Angle interrupt 10
DSSINTSMC_ANGLE_INT_11_NO	122	Angle interrupt 11
DSSINTSMC_ANGLE_INT_12_NO	123	Angle interrupt 12
DSSINTSMC_ANGLE_INT_13_NO	124	Angle interrupt 13
DSSINTSMC_ANGLE_INT_14_NO	125	Angle interrupt 14
DSSINTSMC_ANGLE_INT_15_NO	126	Angle interrupt 15
DSSINTSMC_ANGLE_INT_16_NO	127	Angle interrupt 16

End-of-Conversion Interrupt

A/D conversion

The end-of-conversion interrupt (EOC) is provided by the A/D conversion feature. This interrupt can be enabled for standard and burst conversion in single conversion mode. In continuous conversion mode, the EOC interrupt is disabled.The EOC interrupt in single conversion mode shows that a conversion has finished and that new ADC values are available.

Required RTI blocks

Your model must contain the following RTI blocks:

- RPCU_ADC_BLx

  The **Enable interrupt** checkbox in the dialog of the RPCU_ADC_BLx block must be selected.

  Note

  If Trigger source is set to “Sample base rate”, the RPCU_ADC_BLx block must not reside in a subsystem that is triggered by the block’s own EOC interrupt.

- To make this interrupt available in your model, you have to add an RPCU_INTERRUPT_BLx block and select the appropriate interrupt source in the block's dialog.

For details on the RTI blocks, refer to the RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮.

Related topics

Basics

A/D Conversion..39

Angle-Based Interrupt

Angular Computation Unit

The angle-based interrupt is based on the angular computation unit (ACU) of the engine control. It is required for angle-based signal generation and measurement functions. The crankshaft and the camshaft signal measurement must be initialized. That means that your model must contain the RPCU_CRANK_SETUP_TPU_BLx and RPCU_CAM_TPU_BLx block, if you want to use the angle-based interrupt. For basic information, refer to Engine Control on page 135.

There are two modes for angle-based interrupt generation, which you can specify in the block dialog of the RPCU_ANGLE_INT_BLx block:

- Up to 16 fixed angle-based subinterrupts can be registered. These are triggered on a specified angle value during run time.
- Up to 6 periodic angle-based subinterrupts can be specified by a start angle and an angle period. A subinterrupt is triggered after each angle period.

Required RTI blocks

To make this interrupt available in your model, you have to add an RPCU_ANGLE_INT_BLx and an RPCU_INTERRUPT_BLx block and select the appropriate interrupt source in the block's dialog.

Related topics

Basics

Generating Angle-Based Interrupts..263

Camshaft Interrupt

Evaluation of the camshaft signal

The camshaft signal can be used as a trigger source for interrupt generation. Whenever an edge of the camshaft signal is detected, an interrupt is generated.

Camshaft interrupts can be generated as follows:

- On each rising edge of the camshaft signal
- On each falling edge of the camshaft signal
- On each rising and each falling edge of the camshaft signal

Required RTI blocks

Your model must contain the following RTI blocks:

- RPCU_CRANK_SETUP_TPU_BLx
- RPCU_CAM_TPU_BLx

  The Enable interrupt checkbox in the dialog of the RPCU_CAM_TPU_BLx block (CAM Phase Measurement Page (RPCU_CAM_TPU_BLx)) must be selected. However, some restrictions must be considered, as described there.

  Note

  If Synchronization mode is set to “Lock position”, the RPCU_CAM_TPU_BLx block must not reside in a subsystem that is triggered by the block’s own interrupt.

- To make the camshaft interrupt available in your model, you have to add an RPCU_INTERRUPT_BLx block and select the appropriate interrupt source in the block's dialog.

For details on the RTI blocks, refer to the RapidPro System – I/O Subsystem MPC565 RTI Reference 📖.

Related topics

Basics

PWM Interrupt

PWM functions with interrupt generation

Note

Interrupt generation in conjunction with PWM signal generation is only possible using the PWM functions of the TPU. The MIOS PWM feature does not support interrupt generation.

1-phase PWM signal generation

Interrupt generation depends on the specified number of periods after which an interrupt is to be generated. Interrupts are generated on the rising edge.

Multi-channel PWM signal generation

Interrupt generation depends on the specified number of periods and the specified position delay after which an interrupt is to be generated. With edge-aligned multi-channel PWM signal generation, interrupts are generated on the rising edge. With center-aligned multi-channel PWM signal generation, interrupts are generated on the middle of the high time signals.

Required RTI blocks

To make a PWM interrupt available in your model, you have to add an RPCU_INTERRUPT_BLx block and select the appropriate interrupt source in the block's dialog.

Related topics

Basics

Index Found Interrupt

Incremental encoder interface

The RapidPro Control Unit provides interrupts triggered by an encoder index signal. The interrupts are generated according to the specified index mode. The index signal can be used to reset the encoder position to an initial value. Reset can be done only at the first index signal or at each index signal. If you have enabled interrupt generation, an interrupt will be send to the master, if the position is reset.

Required RTI blocks

To make this interrupt available in your model, you have to add an RPCU_INTERRUPT_BLx block and select the appropriate interrupt source in the block's dialog.

Related topics

Basics

Incremental Encoder Interface.. 103

Serial Data Interrupt

Knock signal measurement

A serial data interrupt can be generated and is sent to the master each time knock signal measurement stops and the measurement result is written to the buffer of the specified serial receiver on the RapidPro Control Unit.

You can use the interrupt to trigger the reading of the buffer, as described in How to Implement the Reading of the Measurement Results on page 333.

Required RTI blocks

To make this interrupt available in your model, you have to add an RPCU_INTERRUPT_BLx block and select the SERDAT interrupt source in the block's dialog.

Related topics

Basics

Event-Triggered Reading of the Measurement Results... 329

Data Ready Interrupt

Measurement results

When request-read is disabled a data ready interrupt is generated and sent to the master each time new measurement values exist in the DPMEM of the RapidPro Control Unit:

- PWM Signal Measurement (PWM2D)
  - RPCU_PWM2D_TPU_BLx
- Pulse Width Measurement (PW2D)
  - RPCU_PW2D_MIOS_BLx
  - RPCU_PW2D_TPU_BLx
- Frequency Measurement (F2D)
  - RPCU_F2D_MIOS_BLx
  - RPCU_F2D_TPU_BLx
- Bit I/O
  - RPCU_BIT_IN_TPU_BLx

You can use the interrupt to trigger the reading of the DPMEM.

Required RTI blocks

To make this interrupt available in your model, you have to add an RPCU_INTERRUPT_BLx block and select the appropriate interrupt source in the block's dialog.

System Functions

Objective

The system functions of the RapidPro Control Unit are used for power management of the installed RapidPro Units and modules within a stack.

Where to go from here

Information in this section

Information in other sections

System (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
Gives information about the RTI block concerned with the power management of the installed RapidPro Units and modules within a stack.

System (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)
Gives information about the RTLib functions concerned with the power management of the installed RapidPro Units and modules within a stack.

Basics on System Functions

Objective

With the power management facility, you can manage the sleep mode of the RapidPro system.

Behavior of the RapidPro system in sleep mode

When you activate the sleep mode of the RapidPro Control Unit, the Control Unit's power is reduced according to the specified power save mode (see below). Optionally, you can also switch all other units in your RapidPro system to standby mode. The Control Unit's microprocessor stops executing program code. All the outputs of the RapidPro system are disabled. All signals which are connected to the RapidPro system are ignored.

Note

You must guarantee that your model or application does not execute an RTI RPCU block or RapidPro Control Unit RTLib function during an activated sleep mode. The consequence will be a task overrun when you wake up the system.

Behavior on wake-up

The master can wake up the slave. The units are reset from standby mode and all the outputs are re-enabled. Software execution from the slave application resumes from the sleep position. There is no re-initialization of the output signals. The time which is needed to wake up the RapidPro system depends on the low power mode.

Power save modes

There are three different power save modes which are supported by the MPC565 microprocessor:

- Doze

  The microcontroller is switched to standby mode.

  Note

  This mode is not supported by RTI.

- Sleep

  In addition, the clock sources are switched to standby mode (for example decrementer and PPC real-time clock).
- Deep Sleep

  In addition, the system clock is switched to standby mode.

Switching off MicroAutoBox II and RapidPro I/O subsystem

If you want to ensure that a combination of MicroAutoBox II and RapidPro I/O subsystem powers down after the termination tasks of all the blocks in the master application are finished, you can use the DS1401_POWER_DOWN block in the same way as it is possible for a MicroAutoBox II alone. For details, refer to DS1401_POWER_DOWN (MicroAutoBox II RTI Reference 📖).

Note

The *Remote In* hardware input pins (KL15) of the MicroAutoBox II and the RapidPro I/O subsystem must be connected.

RTI/RTLib support

You can access the system features via the RTI RPCU Blockset and RTLib functions. For details, see:

- System (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮)
- System (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮)

Related topics

References

DS1401_POWER_DOWN (MicroAutoBox II RTI Reference 🕮)

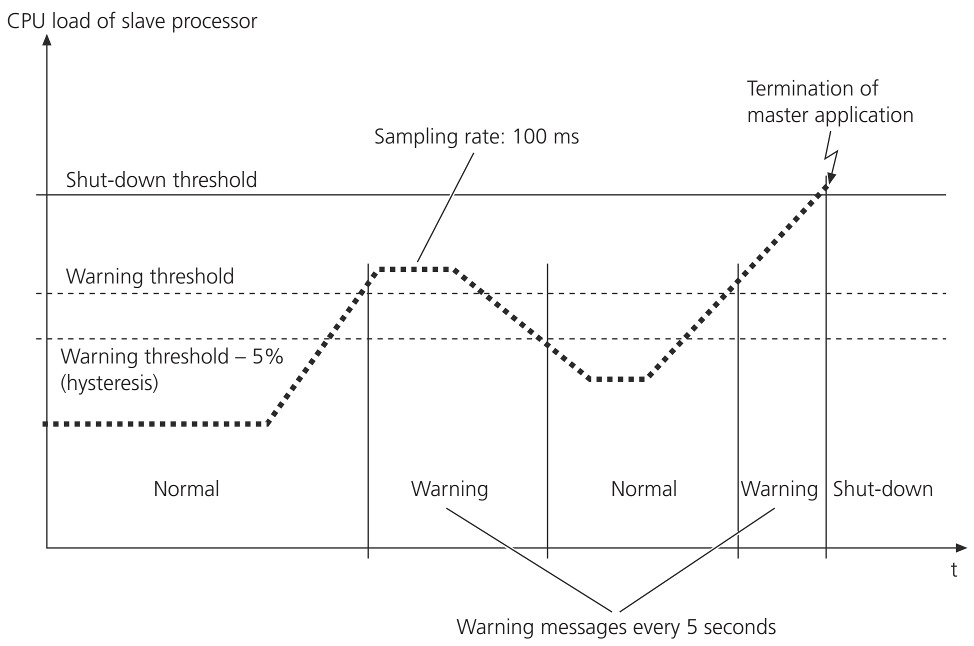
CPU Load Measurement of the Slave Processor

Objective

The CPU load of the MPC565 is monitored and automatic shutdown of the master application running on MicroAutoBox II or DS1007 can be triggered if an overload occurs.

CPU load measurement

After successful initialization of the RapidPro system, the CPU load (in %) of the MPC565 is measured every 100 ms, and the results are sent to the RCP system via the ECU interface. The CPU load is evaluated relative to a shutdown threshold and (optionally) to a warning threshold (only RTI). Depending on this evaluation, your RapidPro system can have three operating states: Normal, warning, and shutdown.



Note

When an RTI model is running the measurement and evaluation of the CPU load of the MPC565 are permanently active. You cannot deactivate it.

Warning threshold

If the CPU load of the MPC565 exceeds this user-defined threshold the RapidPro system generates a warning message every 5 seconds as long as the system stays in the "Warning" state. You can specify whether a warning threshold is active, refer to Parameter Page (RPCU_SETUP_BLx). Additionally, you can activate a status flag (0: "Normal" state, 1:"Warning" state) that can be used in your RTI model, for example, as trigger source.

Note

The RapidPro system switches from "Warning" state back to "Normal" state not before the CPU load becomes less than *WarningTreshold - 5%* (hysteresis).

Shutdown threshold

If the CPU load of the MPC565 exceeds this user-defined threshold the RapidPro system sets the simState variable to STOP:All measurement functions of the RapidPro system are stopped and the master application running on MicroAutoBox II or DS1007 sends the termination values (if defined) for the according output functions to the RapidPro system. If no termination values are defined the output functions work with the last-measured data. In consequence, the master application stops. For details on specifying a shutdown threshold, refer to Parameter Page (RPCU_SETUP_BLx).

RTI/RTLib support

You can access this system feature via the RTI RPCU Blockset and RTLib functions. For details, see:

- Parameter Page (RPCU_SETUP_BLx) (RapidPro System – I/O Subsystem MPC565 RTI Reference 🕮).
- CPU Load Measurement (RapidPro System – I/O Subsystem MPC565 RTLib Reference 🕮).

Application Examples

Objective	The following application examples give you detailed instructions on using the RapidPro Control Unit Blockset.
Demo models	The RapidPro Control Unit Blockset contains several demo models, which you can access via the **Demos** button in the blockset.

Where to go from here

Information in this section

Application Examples for A/D Conversion

Objective

Analog signals can be converted into digital signals by using an A/D conversion block in your Simulink model. The following instructions show you how to specify the RPCU_ADC_BLx block for different purposes.

Demo models

The following demo models of the RapidPro Control Unit Blockset show the use of ADC RTI blocks:

- DemoRPCUPWMTrigger (Standard A/D conversion on one channel, triggered by a PWM interrupt)
- DemoQADCBurst (Burst A/D conversion on two channels, triggered by a fixed angle-based interrupt)

Where to go from here

Information in this section

How to Use Standard A/D Conversion

Objective

The following instructions show you basically how to configure the block for standard A/D conversion.

Preconditions

The RapidPro system must contain modules that support analog signals. These can be signal conditioning modules of type SC-AI x/y or any other module providing analog signals.

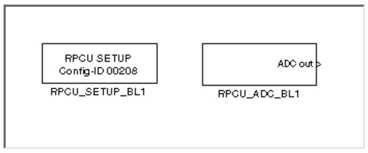
Method

To use standard A/D conversion

1 Open a new Simulink model.

2 Drag an RPCU_SETUP_BLx block into the model to load the hardware topology file for RTI (see also How to Import the Hardware Topology to a Model on page 25).

3 Drag an RPCU_ADC_BLx block into the model and connect it to another Simulink block.



4 Double-click the RPCU_ADC_BLx block to open its dialog.

5 Specify the settings of the block.

- The settings on the **Unit** page depend on the connected RCP system.
- You can select at most the number of analog channels that are provided by the installed modules. The available channels are displayed on the **Parameters** page.
- If you select the Engine mode as the trigger source, there must be an RPCU_ANGLE_INT_BLx block in the model. The angle-based interrupt requires further blocks in the model (RPCU_TIMER_SETUP_TPU_BLx, RPCU_CRANK_SETUP_TPU_BLx, RPCU_CAM_TPU_BLx).
- If you select the PWM mode as trigger source, there must be a TPU_PWM block in the model.
- The read mode can only be selected if you have specified standard conversion with the **Sample base rate** as the trigger source and **Single** as the conversion mode.
- If you enable the status outport on the **Advanced** page, the **Status** outport is added to the block. The value of the status signal shows you whether the value in the buffer has been already read (old value) or has been updated since the last read (new value).
- If you select the Enable interrupt checkbox on the Advanced page, you make the end-of-conversion interrupt available to the RPCU_INTERRUPT_BLx block.

6 Click **OK** to close the dialog.

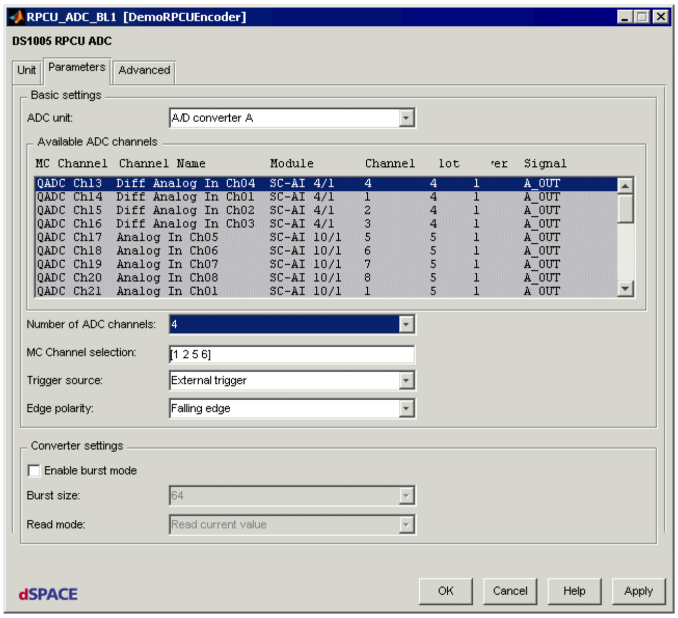
If there are wrong settings or conflicts with other channels, an error message will appear.

Result

The model is now prepared for executing a standard A/D conversion with the specified settings.

Example

The following settings perform an A/D conversion on converter A with 4 channels (channels 1, 2, 5 and 6). The conversion start is triggered by the falling edge of an external signal.



Basic settings

ADC unit: A/D converter A

Available ADC channels

MC Channel	Channel Name	Module	Channel	lot	'er	Signal
QADC Ch13	Diff Analog In Ch04	SC-AI 4/1	4	4	1	A_OUT
QADC Ch14	Diff Analog In Ch01	SC-AI 4/1	1	4	1	A_OUT
QADC Ch15	Diff Analog In Ch02	SC-AI 4/1	2	4	1	A_OUT
QADC Ch16	Diff Analog In Ch03	SC-AI 4/1	3	4	1	A_OUT
QADC Ch17	Analog In Ch05	SC-AI 10/1	5	5	1	A_OUT
QADC Ch18	Analog In Ch06	SC-AI 10/1	6	5	1	A_OUT
QADC Ch19	Analog In Ch07	SC-AI 10/1	7	5	1	A_OUT
QADC Ch20	Analog In Ch08	SC-AI 10/1	8	5	1	A_OUT
QADC Ch21	Analog In Ch01	SC-AI 10/1	1	5	1	A_OUT

Number of ADC channels: 4

MC Channel selection: [1 2 5 6]

Trigger source: External trigger

Edge polarity: Falling edge

Converter settings

☐ Enable burst mode

Burst size: 64

Read mode: Read current value

How to Use Burst A/D Conversion

Objective

Using the RapidPro Control Unit, you can convert analog signals in burst conversion mode to get the shape of an input signal.

Preconditions

The RapidPro system must contain modules that support analog signals. These can be signal conditioning modules of type SC-AI x/y or any other module providing analog signals.

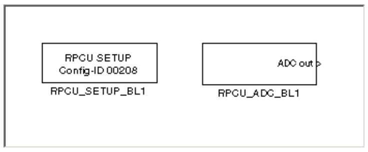
Method

To use burst A/D conversion

1 Open a new Simulink model.

2 Drag an RPCU_SETUP_BLx block into the model to load the hardware topology file for RTI (see also How to Import the Hardware Topology to a Model on page 25).

3 Drag an RPCU_ADC_BLx block into the model and connect it to another Simulink block.



4 Double-click the RPCU_ADC_BLx block to open its dialog.

5 Select the **Enable burst mode** checkbox on the **Parameters** page to activate a burst conversion. The **Channel** signal is added to the block to index the channel for which the last conversion burst was performed in the channel list. The channel number represents the position within the specified channel list, not the channel number itself.

6 Select the required **Burst size** from the drop-down list. This is the number of conversions that are executed on one channel in the channel list, if the A/D converter was triggered to start.

7 Specify the other settings of the block.
   - The settings on the **Unit** page depend on the connected RCP system.
   - You can select at most the number of analog channels that are provided by the installed modules. The available channels are displayed on the **Parameters** page.
   - If you select the Engine mode as the trigger source, there must be an RPCU_ANGLE_INT_BLx block in the model. The angle-based interrupt requires further blocks in the model (RPCU_TIMER_SETUP_TPU_BLx, RPCU_CRANK_SETUP_TPU_BLx, RPCU_CAM_TPU_BLx).
   - If you select the PWM mode as trigger source, there must be a TPU_PWM block in the model.
   - The read mode can only be selected if you have specified standard conversion with the **Sample base rate** as the trigger source and **Single** as the conversion mode.
   - If you enable the status outport on the **Advanced** page, the **Status** outport is added to the block. The value of the status signal shows you whether the value in the buffer has been already read (old value) or has been updated since the last read (new value).
   - If you select the Enable interrupt checkbox on the Advanced page, you make the end-of-conversion interrupt available to the RPCU_INTERRUPT_BLx block.

8 Click **OK** to close the dialog.

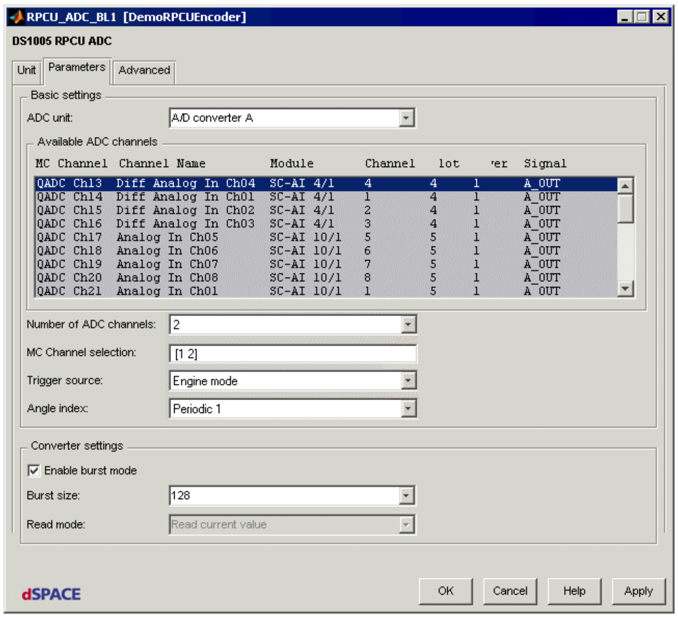
   If there are wrong settings or conflicts with other channels, an error message will appear.

Result

The model is now prepared for executing a burst A/D conversion with the specified settings.

Example

The following settings perform a burst A/D conversion with a burst size of 128 on converter A with 2 channels (channels 1 and 2). The conversion start is triggered by a periodic angle interrupt.



RPCU_ADC_BL1 [DemoRPCUEncoder]

DS1005 RPCU ADC

Basic settings

ADC unit: A/D converter A

Available ADC channels

MC Channel	Channel Name	Module	Channel	lot	'er	Signal
QADC Ch13	Diff Analog In Ch04	SC-AI 4/1	4	4	1	A_OUT
QADC Ch14	Diff Analog In Ch01	SC-AI 4/1	1	4	1	A_OUT
QADC Ch15	Diff Analog In Ch02	SC-AI 4/1	2	4	1	A_OUT
QADC Ch16	Diff Analog In Ch03	SC-AI 4/1	3	4	1	A_OUT
QADC Ch17	Analog In Ch05	SC-AI 10/1	5	5	1	A_OUT
QADC Ch18	Analog In Ch06	SC-AI 10/1	6	5	1	A_OUT
QADC Ch19	Analog In Ch07	SC-AI 10/1	7	5	1	A_OUT
QADC Ch20	Analog In Ch08	SC-AI 10/1	8	5	1	A_OUT
QADC Ch21	Analog In Ch01	SC-AI 10/1	1	5	1	A_OUT

Number of ADC channels: 2

MC Channel selection: [1 2]

Trigger source: Engine mode

Angle index: Periodic 1

Converter settings

[✓] Enable burst mode

Burst size: 128

Read mode: Read current value

Application Examples for Bit I/O

Objective

The RapidPro Control Unit Blockset provides RTI blocks for handling digital input and output signals bitwise. The following instructions show you how to use the bit I/O blocks implemented on the TPU and on the I/O PLD.

Where to go from here

Information in this section

How to Handle Digital Read Access Using a TPU

Objective

Digital input signals can be made available to your Simulink model by using the bit I/O blocks of the RTI RPCU blockset. The following instructions show you how to select and parameterize the read blocks for a TPU.

Preconditions

The RapidPro system must contain modules that provide digital input signals. These can be signal conditioning modules of type SC-DI 8/1 or any other module with digital input channels.

Note

Check the state of the I/O pins before you connect electronic devices to your system.

Method

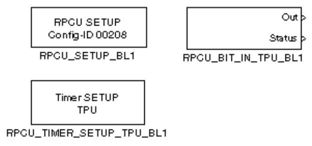
To handle digital read access using a TPU

1 Drag an RPCU_SETUP_BLx block into the model to load the hardware configuration file (see How to Import the Hardware Topology to a Model on page 25).

2 Drag an RPCU_TIMER_SETUP_TPU_BLx block into the model to specify the clock source and the resolution of the TPU timer (see How to Specify TPU Timers on page 56).

The settings of this block have no affect on the bit I/O block.

3 Drag an RPCU_BIT_IN_TPU_BLx block into the model and connect it to another Simulink block.



4 Open the RPCU_BIT_IN_TPU_BLx block and specify its parameters:
   - Select a channel from the channel list on the **Unit** page.
   - Select the Enable interrupt checkbox on the Parameters page and the edge polarity to be detected, if you want to make an interrupt available in the RPCU_INTERRUPT_BLx block.

5 Click **OK** to close the dialog.

   If there are wrong settings or conflicts with other channels, an error message will be generated.

Result

The model is prepared for reading digital input signals from the specified channels.

How to Handle Digital Signals Using the I/O PLD

Objective

Digital input and output signals can be made available to your Simulink model by using the bit I/O blocks of the RPCU blockset. The following instructions show you how to select and parameterize the read and write blocks for the I/O PLD.

Preconditions

The RapidPro system must contain modules that provide digital input or output signals. These can be signal conditioning modules of type SC-DI 8/1 or SC-DO 8/1, or any other module with digital input or output channels. If your RapidPro system contains a Power Unit, you can also use PS modules for digital outputs.

Note

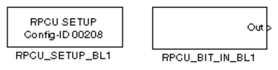
Check the state of the I/O pins before you connect electronic devices to your system.

Method

To handle digital signals using the I/O PLD

1 Drag an RPCU_SETUP_BLx block into the model to load the hardware configuration file (see How to Import the Hardware Topology to a Model on page 25).

2 Drag an RPCU_BIT_IN_BLx block into the model for read access to digital input signals or an RPCU_BIT_OUT_BLx block for write access to digital output signals. The blocks must be connected to other Simulink blocks.



3 Open the RPCU_BIT_IN_BLx block or the RPCU_BIT_OUT_BLx block and specify the block settings:

- Select the group from which you want to use the channels on the **Parameters** page.
- For "Bit In": Select the **Read mode** on the **Advanced** page.

  For "Bit Out": Specify the initialization and termination values on the corresponding dialog pages.

4 Click **OK** to close the dialog.

If there are wrong settings or conflicts with other channels, an error message will appear.

Result

The model is prepared for reading digital input signals from the specified channels and writing digital output signals to the specified channels.

Application Examples for Timing I/O

Objective

The 1-phase PWM signal generation can be implemented on a TPU, but also on the MIOS. The following instructions show you how to implement 1-phase PWM signal generation in your Simulink model.

Demo models

The following demo model of the RapidPro Control Unit Blockset demonstrates the use of PWM RTI blocks:

- **DemoRPCUPWMTrigger** A TPU PWM block for 1-phase PWM signal generation is used to trigger a standard A/D conversion.

Where to go from here

Information in this section

How to Generate 1-Phase PWM Signals on the MIOS

Preconditions

The RapidPro Control Unit must contain modules that provide digital output signals. These can be signal conditioning modules of type SC-DO x/y, or any other module with digital output channels. If your RapidPro system contains a Power Unit, you can also use PS modules for digital outputs.

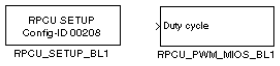
The hardware configuration must suit the connected actuators or power stages and the blocks's settings.

Method

To generate a 1-phase PWM signal on the MIOS

1 Drag an RPCU_SETUP_BLx block into the model to load the hardware configuration file (see How to Import the Hardware Topology to a Model on page 25).

2 Drag an RPCU_PWM_MIOS_BLx block into the model.



3 Connect the inports with other Simulink blocks.

4 Open the RPCU_PWM_MIOS_BLx block and specify its parameters:
   - Select a channel from the channel list on the **Unit** page.
   - Select the most suitable period range for the PWM signal you want to generate.
   - Specify the initialization and termination values on the **Parameters** page.
   - Select the **Enable synchronous start** checkbox to synchronize the start of multiple MIOS PWM signals in your model.
   - Select the **Enable inport Period** checkbox, if you want to update the period during run time. The inport must be connected to another Simulink.

5 Click **OK** to close the dialog.

   If there are wrong settings or conflicts with other channels, an error message will appear.

Result

Your model is prepared for generating a 1-phase PWM signal on the MIOS.

How to Generate a 1-Phase PWM Signal on the TPU

Preconditions

The RapidPro Control Unit must contain modules that provide digital output signals. These can be signal conditioning modules of type SC-DO x/y, or any other module with digital output channels. If your RapidPro system contains a Power Unit, you can also use PS modules for digital outputs.

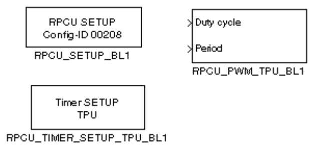
Method

To generate a 1-phase PWM signal on the TPU

1 Drag an RPCU_SETUP_BLx block into the model to load the hardware configuration file (see How to Import the Hardware Topology to a Model on page 25).

2 Drag an RPCU_TIMER_SETUP_TPU_BLx block into the model to specify the clock source and the resolution of the TPU timer (see How to Specify TPU Timers on page 56).

3 Drag an RPCU_PWM_TPU_BLx block into the model.



4 Open the RPCU_PWM_TPU_BLx block and specify its parameters:
   - Select a channel from the channel list on the **Unit** page.
   - Select a timer to be used and specify its period range. The selectable period range depends on the timer resolution which you previously specified in the RPCU_TIMER_SETUP_TPU_BLx block.
   - Specify the initialization and termination values on the **Parameters** page.

- Select the Enable interrupt checkbox and enter the number of periods after which an interrupt is to be generated, if you want to make a PWM interrupt available in the RPCU_INTERRUPT_BLx block.

5 Click OK to close the dialog.

If there are wrong settings or conflicts with other channels, an error message will appear.

Result

Your model is prepared for generating a 1-phase PWM signal on a TPU.

How to Measure Speed and Position of an Incremental Encoder

Preconditions

The RapidPro system must contain modules that provide digital input signals. These can be signal conditioning modules of type SC-DI 8/1 or any other module with digital input channels. Processing the signal of an incremental encoder requires three consecutive TPU input channels.

Note

Check the state of the I/O pins before you connect electronic devices to your system.

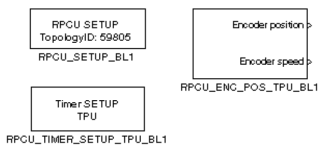
Method

To measure the speed and position of an incremental encoder

1 Drag an RPCU_SETUP_BLx block into the model to load the hardware configuration file (see How to Import the Hardware Topology to a Model on page 25).

2 Drag an RPCU_TIMER_SETUP_TPU_BLx block into the model to specify the clock source and the resolution of the TPU timer (see How to Specify TPU Timers on page 56).

3 Drag an RPCU_ENC_POS_TPU_BLx block into the model.



4 Open the RPCU_ENC_POS_TPU_BLx block and select a channel from the channel list on the **Unit** page. In addition, the two successive channels are selected automatically.

5 Click **OK** to close the dialog.

If there are wrong settings or conflicts with other channels, an error message will be generated.

6 Connect the outports of the RPCU_ENC_POS_TPU_BLx block to non-virtual blocks, for example, Gain blocks (K = 1). The RPCU_ENC_POS_TPU_BLx block would otherwise be neglected during code generation due to code optimization.

Result

The model is prepared for measuring the speed [1/s] and the position [-8192.00 … +8191.75] of an incremental encoder.

Example

In the **DemoRPCUEncoder** demo model of the RapidPro Control Unit Blockset, two encoder signals are measured:

- The RPCU_ENC_POS_TPU_BL1 block generates an interrupt at each occurrence of an index signal. This interrupt triggers a subsystem that counts the number of detected index signals.
- The RPCU_ENC_POS_SET_TPU_BL1 block is able to set the incremental encoder position value of RPCU_ENC_POS_TPU_BL2 to 250.

Application Examples for Engine Control

Where to go from here

Information in this section

Example of Crankshaft and Camshaft Signals

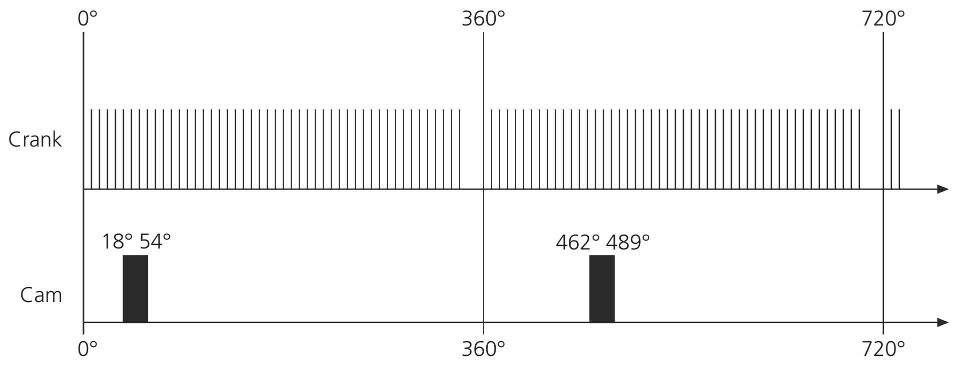
Use scenario

This example shows the characteristics of synchronized crankshaft and camshaft signals, and the relationship between crankshaft and camshaft wheels and signal output.

Suppose you have the following crankshaft and camshaft wheels:

Wheel	Parameters
Crankshaft	60 – 2 wheel: ▪ 60 teeth ▪ One gap with two missing teeth (segment = 360°) Edge polarity: "Rising edges"
Camshaft	Markers: ▪ #1: 18° … 54° (1st crankshaft revolution) ▪ #2: 462° … 489° (2nd crankshaft revolution)

The following crankshaft and camshaft signals are generated, and the 0° position equals the first rising edge of the crankshaft signal:

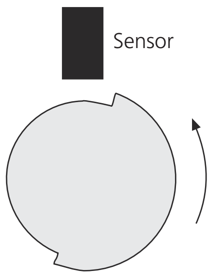
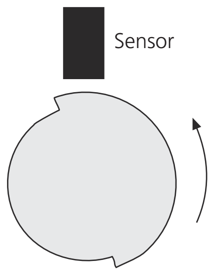
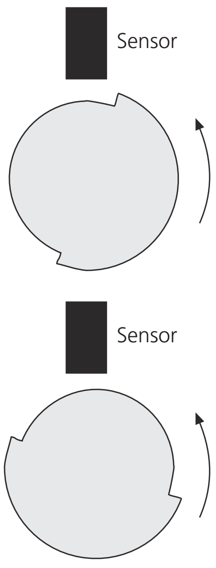
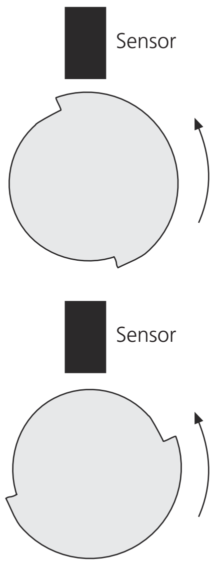


For the crankshaft signal, one peak represents one tooth. For the camshaft signal, one peak represents one camshaft marker.

Example of Synchronization with Multiple Camshaft Wheels

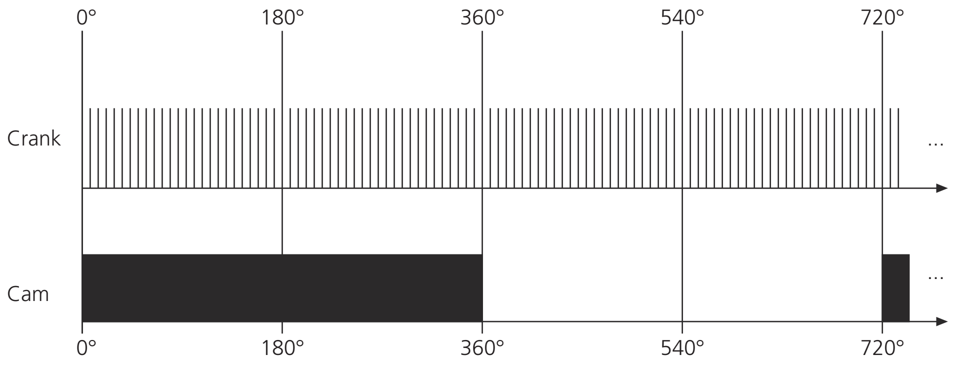
Camshaft wheels

Suppose you have an engine with one camshaft wheel, and another engine with two camshaft wheels. The crankshaft wheels have no gaps. Each camshaft marker comprises an angle range of 360°. For the engine with the two camshaft wheels, the wheels are arranged orthogonally. The polarity of the camshaft signal is set to "Either edges".

Number of Camshaft Wheels	**Time Required for Synchronization:**	
	Best Case	**Worst Case**
One	 immediately	 after about a half camshaft revolution
Two	 immediately (signal 1)	 after about a fourth camshaft revolution (signal 2)

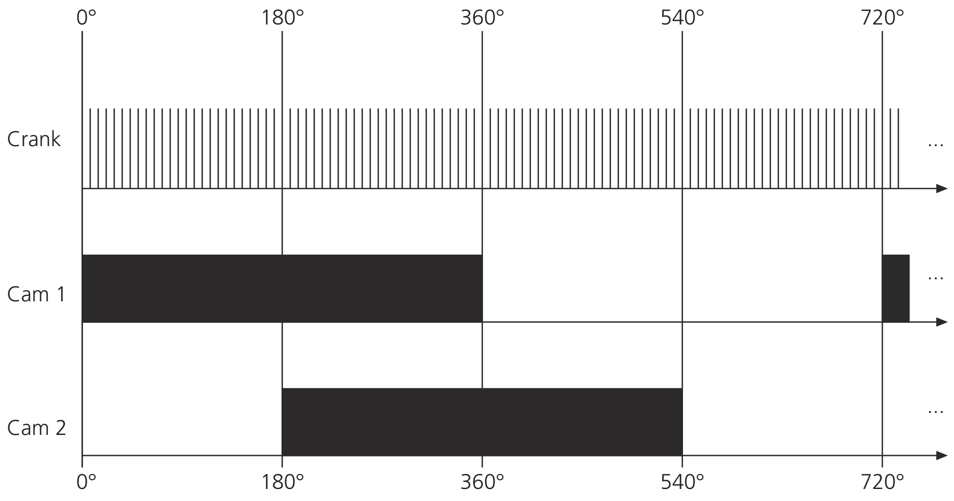
Signals of one camshaft wheel

The following crankshaft and camshaft signals are generated in the case of one camshaft wheel:



Signals of two camshaft wheels

The following crankshaft and camshaft signals are generated in the case of two camshaft wheels:



Synchronization time

In the best case, for both scenarios, initialization is achieved immediately. In the worst case, however, the engine with one camshaft wheel is initialized after about one crankshaft revolution (half a camshaft revolution), but the engine with the two camshaft wheels is initialized after about half a crankshaft revolution (fourth camshaft revolution).

Related topics

Examples

Example of the Benefit of a Camshaft Wheel with Multiple Camshaft Markers........................ 179

DemoRPCUEngineControl Model

Use scenario

The DemoRPCUEngineControl model of the RapidPro Control Unit Blockset shows the use of the engine control RTI blocks:

- Injection and ignition pulse generation for a 6-cylinder engine
- Pulse update triggered by 6 fixed angle-based interrupts

Method

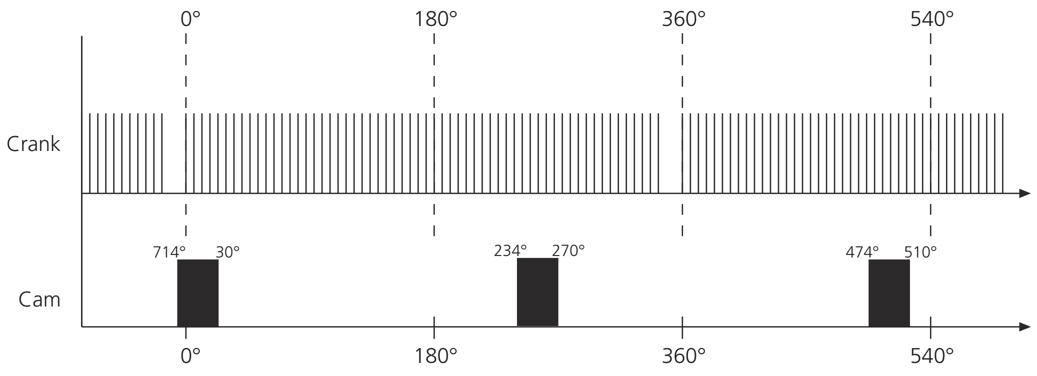
To open the DemoRPCUEngineControl model

1 Enter **`rtirpcudemolib`** in the MATLAB Command Window.
  The Library: rtirpcudemolib window opens.
2 Double-click the Engine Control button.
  The `DemoRPCUEngineControl` model opens.

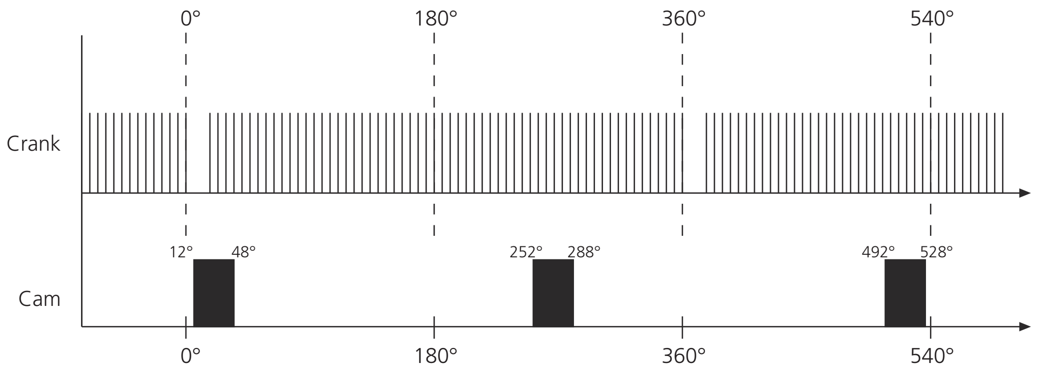
Example of Implementing an Engine Offset

Use scenario

Suppose, the following crankshaft and camshaft signals are generated:



Engine offset Suppose you prefer the last edge of the crankshaft signal before the gap to be the 0° position instead of the first edge after the gap. In consequence, the angle axes must be shifted 18° to the left (i.e., engine offset = +18°):



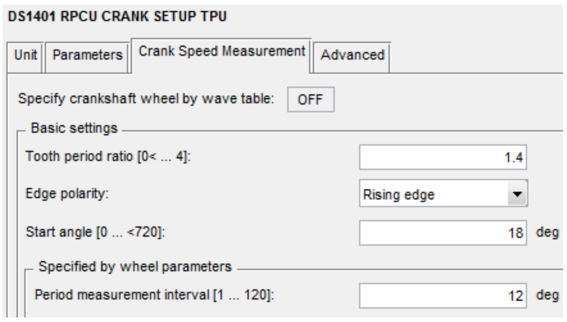
Note

In the above example, the angle shift (engine offset) causes the following:

- Change of the camshaft signal polarity at 0°, from "High" to "Low" (affects the Signal polarity before first transition parameter).
- The order of the rising edges of the camshaft markers changes: 12°, 252°, 492° instead of 234°, 474°, 714°, which affects the specification of the Vector of camshaft start angles parameter.

An implementation approach

In the dialog of the RPCU_CRANK_SETUP_TPU_BLx block, the following settings are chosen to implement the engine offset. The Start angle is determined by the engine offset (18°).



Specify crankshaft wheel by wave table: OFF

Basic settings

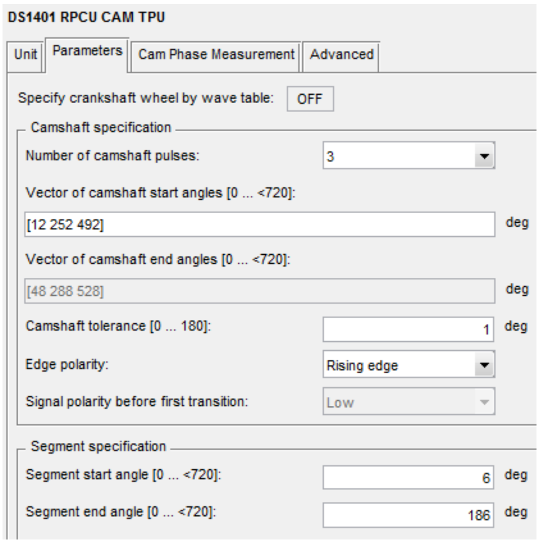
Tooth period ratio [0< ... 4]:	1.4
Edge polarity:	Rising edge
Start angle [0 ... <720]:	18 deg

Specified by wheel parameters

Period measurement interval [1 ... 120]:	12 deg

For the implementation instructions, refer to How to Set Up Engine Offsets on page 209.

In the dialog of the RPCU_CAM_TPU_BLx block, the following settings are chosen to implement the engine offset. The Signal polarity before first transition is not evaluated as the Edge polarity parameter is *not* set to "Either edge". The Segment start angle and Segment end angle parameters affect the time it takes to synchronize the angle measurement, for details refer to Logging the Camshaft Signal Within the Evaluation Segment on page 162.


DS1401 RPCU CAM TPU

Specify crankshaft wheel by wave table: OFF

Camshaft specification

Number of camshaft pulses: 3

Vector of camshaft start angles [0 ... <720]:

[12 252 492] deg

Vector of camshaft end angles [0 ... <720]:

[48 288 528] deg

Camshaft tolerance [0 ... 180]: 1 deg

Edge polarity: Rising edge

Signal polarity before first transition: Low

Segment specification

Segment start angle [0 ... <720]: 6 deg

Segment end angle [0 ... <720]: 186 deg

With the given crankshaft and camshaft settings, synchronization could take place as follows:

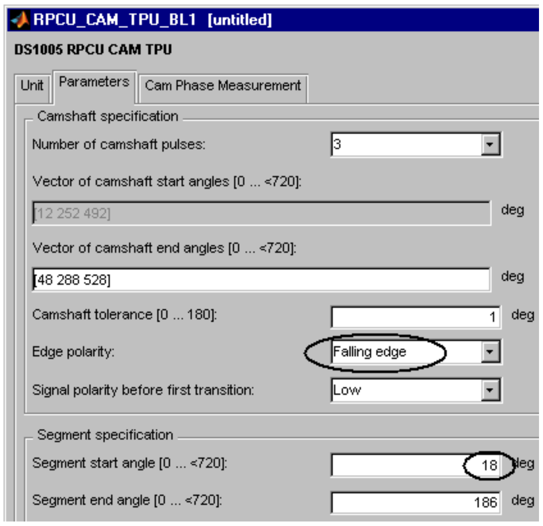
- If the engine starts revolving shortly before the 0° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which is greater than the 6° segment start angle. Thus, the logging of the camshaft signal is not started and synchronization is not possible. However, the angle counter is still running. The 6° segment start angle is reached again after about two engine revolutions (18°+59*12°=726°=6°). Synchronization is finished after about 900°, determined by the segment end angle.
- If the engine starts revolving shortly before the 360° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which is greater than the 6° segment start angle. Thus, the logging of the camshaft signal is not started and synchronization is not possible. However, the angle counter is still running. The 6° segment start angle is reached again after about two engine revolutions (726°=6°). Synchronization is finished after about 900°, determined by the segment end angle.

An improved implementation

In the RPCU_CAM_TPU_BLx block dialog, the **Edge polarity** is changed to "Falling edge", and the segment start angle is changed to 18°. All the other settings remain unchanged:



DS1005 RPCU CAM TPU

Camshaft specification

Number of camshaft pulses:	3	
Vector of camshaft start angles [0 ... <720]:	[12 252 492]	deg
Vector of camshaft end angles [0 ... <720]:	[48 288 528]	deg
Camshaft tolerance [0 ... 180]:	1	deg
Edge polarity:	Falling edge	
Signal polarity before first transition:	Low	

Segment specification

Segment start angle [0 ... <720]:	18	deg
Segment end angle [0 ... <720]:	186	deg

With the given settings synchronization can take place as follows:

- If the engine starts revolving shortly before the 0° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which equals the 18° segment start angle. Thus, the logging of the camshaft signal is started. A falling edge at 48° is logged. This matches the marker specification (falling edge at 48°). After the segment end angle (186°) is reached, synchronization is successfully finished.

- If the engine starts revolving shortly before the 360° position:

  The angle measurement starts at the first rising edge of the crankshaft signal after the gap. The start value is 18°, which equals the 18° segment start angle. Thus, the logging of the camshaft signal is started. A falling edge at 168° is logged. This matches the marker specification (falling edge at 528°=168°+360° due to segmentation, refer to Updating the Angle Counter with the Camshaft Marker Specification on page 164). After the segment end angle (186°) is reached, synchronization is successfully finished.

Tip

Synchronization can be efficient in the following case:

```
Start angle ≤ Segment start angle ≤ Min relevant camshaft angle
```

In the above example: 18° ≤ 18° ≤ 48°. It was helpful to change the camshaft edge polarity to "Falling edge".

Note

The Segment start angle must always read as follows:

```
Segment start angle = Start angle + N * Period measurement angle
```

Related topics

HowTos

How to Set Up Engine Offsets.. 209