Control Loop Library

The “Control Loop” library includes controls, filters, and transfer functions for process control and signal processing.

Product description

The “Control Loop” library includes function blocks for process control. It consists of different elements that are assembled flexibly into a control structure. These include various differentiators and integrators, for which various anti-windup strategies are available. In this way, the structure of the created control can be derived from the selected components. Some transfer functions are also achieved according to this scheme. This library also provides an example for the use of nested object-oriented function blocks.

The library also contains several digital filters that can be used for signal processing. For this purpose, it is necessary to determine the rating parameters yourself.

The following function blocks are included in the library:

Controls

- `Controller_Base`: Abstract base function block for creating individual controls
- `Controller_P`: Function block for implementing a P control
- `Controller_PD`: Function block for implementing a PD control
- `Controller_PI`: Function block for implementing a PI control
- `Controller_PID`: Function block for implementing a PID control

Integrator approximation

- `Integrator_Base`: Base function block for creating individual integral approximations
- `Integrator_ParabolicApproximation`: Function block for approximating the integral by means of a parabola using the last values
- `Integrator_RectangleApproximation`: Function block for approximating the integral by means of a rectangle
- `Integrator_TrapezoidApproximation`: Function block for approximating the integral by means of a trapezoid

Anti-windup strategies

These functions blocks provide different strategies to avoid an overflow of the integrator in case of a prolonged control variance.

- `AntiWindUp_Base`: Abstract basis function block for creating individual anti-windup strategies

- `AntiWindUp_Clamping`: Function block for using an anti-windup strategy that fixes the integrator value to the set maximum
- `AntiWindUp_BackCalculation`: Function block for using an anti-windup strategy that decreases the integrator value over time

Differentiator approximation

- `Differentiator_Base`: Abstract basis function block for creating individual differentiator approximations
- `Differentiator_BackwardDifference`: Function block for linear approximation of the integral using the last value
- `Differentiator_LinearAverageApproximation`: Function block for linear approximation of the integral using the last values
- `Differentiator_LinearFourPointApproximation`: Function block for linear approximation of the integral using the last values
- `Differentiator_ParabolicApproximation`: Function block for approximating the differential by means of a parabola using the last values

Two-point controls

- `BangBangController`: Function block for implementing a two-point control
- `BangBangControllerWithTimeHysteresis`: Function block for implementing a two-point control with time-based hysteresis
- `BangBangControllerWithValueHysteresis`: Function block for implementing a two-point control with value-based hysteresis

Three-point controls

- `ThreePointController`: Function block for implementing a three-point control
- `ThreePointControllerWithValueHysteresis`: Function block for implementing a three-point control with value-based modulation

Filters

- `Filter_Base`: Abstract basis function block for creating individual filters
- `Filter_FIR`: Function block for implementing a finite impulse response filter
- `Filter_IIR`: Function block for implementing an infinite impulse response filter
- `Filter_SOS`: Function block for implementing a second-order section filter

PWM generation

- `PWM_CreatorBase`: Abstract basis function block for creating individual PWM signals
- `PWM_Creator`: Function block for creating a PWM signal
- `PWM_Creator_FixedCycle`: Function block for creating a PWM signal whose cycle time may differ from the task cycle time

Transfer functions

- `DT1`: Function block for writing a DT1 transfer member. For easier application, a configuration using the `Differentiator_LinearAverageApproximation` is set as default: `DT1_LinearAverage`
- `IT1`: Function block for writing a IT1 transfer member. For easier application, a configuration using the `Integrator_TrapezoidApproximation` is set as default: `IT1_Trapezoid`
- `PT1`: Function block for writing a PT1 transfer member. For easier application, a configuration using the `Integrator_TrapezoidApproximation` is set as default: `PT1_Trapezoid`
- `PT2`: Function block for writing a PT2 transfer member. For easier application, a configuration using two `Integrator_TrapezoidApproximation` is set as default: `PT2_Trapezoid`

General information

Supplier:

CODESYS GmbH
Memminger Strasse 151
87439 Kempten
Germany

Support:

Technical support is not included with this product. To receive technical support, please purchase a CODESYS Support Ticket.

https://support.codesys.com

Item:

Control Loop Library

Item number:

000113

Sales/Source of supply:

CODESYS Store
https://store.codesys.com

Included in delivery:

CODESYS Package

System requirements and restrictions

Development system	CODESYS Development System V3.5.14.0 or higher
Runtime	CODESYS Control V3.5.14.0 or higher
Supported platforms and devices	Note: Use the “Device Reader” project for locating the functions supported by the PLC. The Device Reader is available in the CODESYS Store free of charge.
Additional requirements	-
Restrictions	-
Licensing	 No license is required.
Required accessories	-

Note: Technical specifications are subject to change. Errors and omissions excepted. The content of the current online version of this document applies.

Creation date: 2024-08-26