Working group MEGMA

Standardization of library blocks
for graphical model exchange

MSR

MEDOC   MEGMA   MEPRO

Version 1.12

Inhaltsverzeichnis

Kapitel 1

General aspects

1.1   Author

Author:	Moser / FKFS at University of Stuttgart
Last changes:	Wednesday, September 19, 2001 at 12:17
Last version:	Version 1.12
Phone:	0049-711-685-8133
E-mail:	moser@fkfs.uni-stuttgart.de

1.2   List of changes

Date	Version	Changes
19/09/2001	1.12	- Correction and adaptation of code and test data for LowpassSecOrd_RE, - Adding additional verbal description to blocks.

Date	Version	Changes
24/08/2001	1.11	- Changed description for round function, - GradientLimiter is now called DifferenceLimiter and new GradientLimiter is added. Both blocks get an additional Enable input, - The general output order is now y, B_max, B_min, - Simulation results are represented as stair plots (no interpolation between points).
05/02/2001	1.10	- Timer_RE: bug on test vectors fixed (R and E true =¿ x unchanged). - Added parenthesis in code for all environments like if or else. - Bug fix of meanvalue code (semicolon replaces comma)
09/10/2000	1.09	- GradientLimiter adapted to other blocks with comparisons. - Hysteresis adapted to other blocks with Enable input. - MaxLogResetEnabled and MinLogResetEnabled with init of B_max and B_min. - Additional descriptions of parameter and constant section. - Added glossary section.

Date	Version	Changes
19/09/2000	1.08	- GradientLimiter gets additional outputs B_min, B_max. - Changed explanation and names of parameters. Added parameter of type logic. - Max/MinLogResetEnabled get additional logic output. - Counters (Counter_RE, Counter_RTE) get another input IV. - One block for Constant and Parameter. - One block Delay_RE replaces both existing versions DelaySignal_RE and DelayValue_RE.
17/08/2000	1.07	Changes defined in the meeting on 16th of August. - Added reset input R and sometimes initial value IV to the blocks DeltaOneStep, DifferenceQuotient, EdgeBi, EdgeFalling, EdgeRising, TurnOnDelayTime, TurnOffDelayTime, TurnOnDelaySample, TurnOffDelaySample, Hysteresis, GradientLimiter. - Limiter, AccumulatorLimited_RE, IntegratorLimited_RE blocks get additional output ports B_min, B_max. - Added additional blocks LeftOpenInterval, RightOpenInterval, OpenInterval in the section Comparison operators and SampleAndHold_RE in the section Miscellaneous. - Section Miscellaneous is deleted and its blocks are distributed into other sections.

Date	Version	Changes
01/08/2000	1.06	Changes defined in the meeting on 26th of July. - Location of the block ports is recommended not defined, - the model exchange is not considered anymore.
2/11/1999	1.05	Front page changed. Version instead of date.
05/08/1999	1.04	1st version distributed to the tool suppliers for implementation of predefined blocks. - Added verbal description for PARAMETER1INDEX and PARAMETER2INDEX, - Added list of 8-10 blocks to be realized by the tool suppliers.
30/07/1999	1.03	- New testdata for turnon/offdelaysample (round(-1.5)=-1), - All blocks with rounding function (e.g. countdown_re, countdown_rte) have corrected graphics.
19/07/1999	1.02	- New document structure - Test cases added and validation of pseudo-code - Meanvalue changed (i¿0)
20/05/1999	1.01	- Added case in Signum function - MeanValue is always calculated over all k elements - Delay-blocks (added else case)
31/03/1999	1.0	Initial version

1.3 Introduction

The MEGMA working group is one of the MSR [1] activities. It's a working group consisting of representatives of ECU-suppliers and car manufacturers like Robert Bosch, Siemens, Hella, Daimler-Chrysler, BMW, Debis and the Research Institute FKFS, which is responsible for the organization and documentation of the results, i.e. this document. The goal of the MEGMA working group is to define a standard library of blocks to be used for typical automotive applications. The standard focuses on blocks with typical ECU[2]-functionality. Engineers recognize these blocks in the different tools because of their identical graphical representation and know the underlying functionality.
These blocks often have typical attributes like:

- discrete time,

- external parameters for adaptation during runtime, i.e. application or

- logical control inputs like Reset and Enable.

In a first step the commitee has defined the functionality of basic blocks with a single rate (one sample time, one task). Aspects like multi-rate systems (multi-tasking), which of course are basic elements of ECU-functionality are considered in the second step.
The basic blocks have been defined by the experts in various sessions and its content is accepted by every participant. A comparison of available blocks has shown, that some basic blocks (e.g. integrator) have to be adapted to functional requests coming from ECU-software (e.g. enable or reset inputs, applicable parameters for limits). By this functional expansions the number of different variants of basic blocks is increasing (e.g. integrator with enable, integrator with enable and reset, ...). All the variants (V1.1, V1.2,...) have the same root, which is called the 'generic' block. The 'generic' block owns all possible functions and therefore is the most general block (e.g. integrator enabled with reset and limits). Only these 'generic' blocks are part of the MSR library. Other variants of any kind of block are expected to be handled by the modeling tool.
Undefined inputs are not allowed. The number of variants in each tool can be different, but each tool supports the MSR- library blocks.
The commitee has left over some important aspects of different blocks, which have to be resolved in agreement of all tool suppliers:

- limit n for the number of inputs of arithmetical or logical operators (MSR: n>2),

- definition of a ϵ as the smallest positive float value greater than zero (used for float comparisons),

---

[1]Manufacturer Supplier Relationship
[2]Electronic Control Unit

- definition of mathematical functions (sin, cos, ...) with respect to the IEEE runtime library.

1.4 Cooperation with tool suppliers

The MEGMA committee has discussed different aspects of a cooperation and integration of various tool suppliers. In these discussions there has been agreement on the following aspects.

1.4.1 Technical issues

The blocks of the MSR-library have to be treated in the different tools with equal functionality offered for the standard libraries of the tools. The list of supported functionality is tool dependent, but often comprises the following functionality:

- production code-generation from the blocks,

- documentation generation,

- simulation (online, offline),

- animation and

- calibration of the blocks.

1.4.2 Legal and business aspects

Both parties (MSR committee and Tool-suppliers) committed on the following aspects:

- MSR-library is treated as any other product of the tool-suppliers (no separate policy),

- the way of implementation is defined by the tool-suppliers under consideration of the technical issues (see section 1.4.1),

- the MSR-library can be publicly accessed,

- no penalty for non-implementation of the library,

- the product can be sold by the tool-suppliers world-wide,

1.5   General definitions for all blocks

The blocks are designed to be used as parts of ECU[3] software and specified for these requirements. Therefore all blocks are discrete time blocks and only have one single sample time dT. The following assumptions refer to the whole MSR-library:

1. Definitions for variables used throughout the MSR-library:

   - Classification of interface variables:
     - Signal inputs I with $I \in \{u, u1, u2, u3, ..., ui\}$,
     - Signal outputs O with $O \in \{y, y1, y2, B\_min, B\_max\}$,
     - Control inputs C with $C \in \{E, R, RT, IV, IV1, IV2, ..., IVi\}$,[4]
     - Parameter inputs P with $P \in \{MX, MN, UMAX, UMIN, LU, LD,$ $LSP, RSP, K, T, m, TD, D, n, l\}$. All parameter inputs can be changed during runtime.

   - Classification of internal variables:
     - State (static) variables: $X \in \{x, x1, x2, x3, ..., xi\}$,
     - Temporary (auto) variables: $T \in \{temp, temp1, temp2\}$,
     - Integer counter variable for loops i,
     - Sample time of the block (time between two successive calls of the run code of the block) dT.

2. Data type definitions:

   - Interface variables can only have two data types: real (64-bit), logic (size not defined, values only 0 or 1).

   - Internal variables (variables only defined within a block) can have an additional integer data type called int (size not defined, signed integer).

3. General graphical representation of each block: The above defined port classifications of the interface variables have recommended positions at each block. Additionally the sequence of all variables in the port classification is recommended, e.g. if a block has parameter inputs MN, MX and K the sequence of the inputs is defined as $P[1] = MX$, $P[2] = MN$, $P[3] = K$. This sequence is the same as the top to down sequence described in the documentation of the variables for each block.

4. A rounding function round() is provided for conversions from real inputs u to integer values. It is defined as

$$round(u) = \lfloor u + 0.5 \rfloor \tag{1.1}$$

---

[3]Electronic Control Unit
[4]E-enable, R-reset, RT-reset trigger, IV-initial value.

Arrows define increasing port numbers
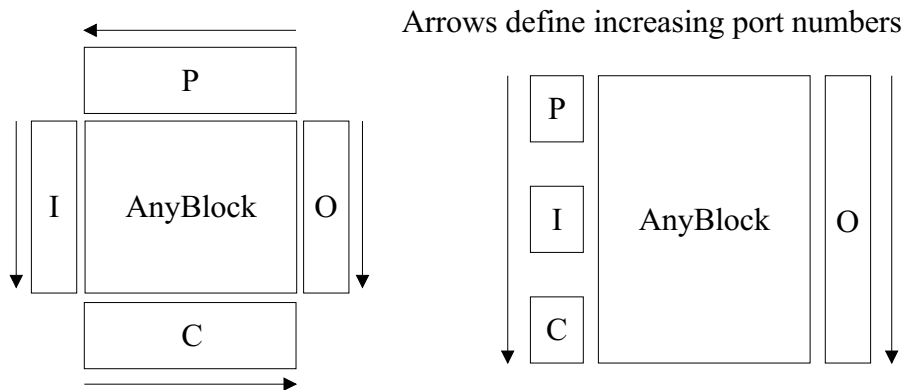
Figure 1.1: Recommendation of the port placement for the different port classifications.

5. Reset functionality: Some blocks have two different ways to reset(see section 2.5 for examples):

   - Static reset input R, i.e. as long as the input remains true the reset is active.

   - Trigger reset input RT, i.e. as long as the input RT is true and was false in the step before, the reset is active.

1.5.1   Model of computation

The computational model defines the execution flow of a system. Common computational models are dataflow (e.g. in Matlab/Simulink) or discrete event (e.g. in ASCET-SD). The dataflow model of computation has constraints for firing (executing), which are derived from input events. That means input data of a block is consumed by executing the block and output data is created. The block-wise execution order of a system is not determined by the graphical model, because there exists no global ordering for the *implicit* events associated with the input data for all blocks in the system. The execution starts at blocks without inputs and blocks with initial state information. A discrete event simulator has a global order for all events, stored in a queue. The *explicit* events include time stamps and are part of the implementation. The simulator executes the events sequentially in time. The goal of the library specification is a description, which is independent of the computational model. Defining functions and a priority for each function for simultaneous activation keeps the execution flow of each block deterministic. The higher the priority the lower the priority value.

1.5.1.1   Execution at multiple rates

To execute a block at multiple, different rates (sample times), the executed code is decomposed into functions. The following functions can be defined for each block:

1. Reset function (reset(), priority 1): Code executed to set the values of state variables of the block,

2. run function (run(), priority 2): Code describing the main functionality of the block,

3. output function (out(), priority 3): Code writing the calculated values on the outputs of the block.

The following rules have to be observed:

- A block with output ports must have at least one output function,

- each function can be called at an arbitrary sample time,

- a function can not be preempted by another function of the same block,

1.5.1.2   Execution with single rate

Single rate is considered as a special case of multiple rates, where all rates are the same. The execution of the function can then be controlled input data, i.e. inputs E, R, RT. Then we only distinguish an initial and a running state. The execution
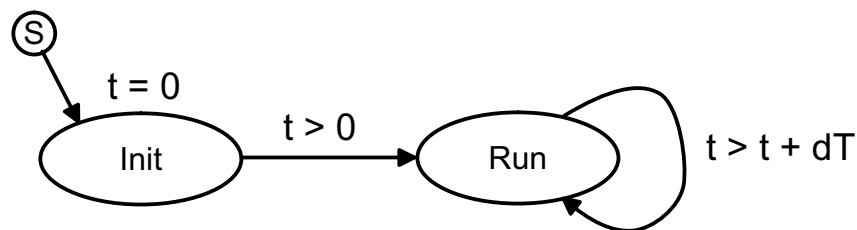
Figure 1.2: Single rate execution for each block of the library.

(see figure 1.2) of a block with a single rate consists of two states:

- Initial state ($t = 0$): Does memory initialization of the state variables (start-up initialization of the ECU).

- Run state ($t > 0$): Recursive call of a run function, which computes the output values out of the input values in each time step defined by the sample time.

The code executed in each of the above mentioned states is defined for each block.

Kapitel 2

Specification of MSR-library

2.1 Arithmetic operators

2.1.1 Division (DIV)

2.1.1.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 u1: real 2 u2: real	1 y: real		

2.1.1.2 Pseudo-code

System-Init-Code	Run-Code
	y = u1 / u2;

2.1.2 Gain (GAIN)

2.1.2.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 u: real	1 y: real		

2.1.2.2 Pseudo-code

System-Init-Code	Run-Code
	y = k * u;

2.1.3   Multiplication (MUL)

2.1.3.1   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
✖	1 ui: real	1 y: real		

2.1.3.2   Pseudo-code

System-Init-Code	Run-Code
	y = u1 * u2 *...* un;

2.1.4   Negation (NEG)

2.1.4.1   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
+/−	1 u: real	1 y: real		

2.1.4.2   Pseudo-code

System-Init-Code	Run-Code
	y = -u;

2.1.5   Sum/Subtraction (SUM)

2.1.5.1   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
SUM	1 ui: real	1 y: real		

2.1.5.2   Pseudo-code

System-Init-Code	Run-Code
	y = u1 - u2 + ... + un;

2.2 Logical operators

2.2.1 AND (AND)

2.2.1.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
AND	1 ui: logic	1 y: logic		

2.2.1.2 Pseudo-code

System-Init-Code	Run-Code
	y = u1 && u2 &&...&& un;

2.2.2 NOT (NOT)

2.2.2.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
NOT	1 u: logic	1 y: logic		

2.2.2.2 Pseudo-code

System-Init-Code	Run-Code
	y = !u;

2.2.3 OR (OR)

2.2.3.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
OR	1 ui: logic	1 y: logic		

2.2.3.2 Pseudo-code

System-Init-Code	Run-Code
	y = u1 || u2 || ... || un;

2.2.4 XOR (XOR)

2.2.4.1 Verbal description

Output is ture, if one of the inputs is true, otherwise the output is false.

2.2.4.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
XOR	1 u1: logic 2 u2: logic	1 y: logic		

2.2.4.3   Pseudo-code

System-Init-Code	Run-Code
	y = (u1 && !u2) || (u2 && !u1);

2.3 Comparison operators

2.3.1 ClosedInterval (CLOSEDINTERVAL)

2.3.1.1 Verbal description

$$y(n) = \left\{ \begin{array}{ccl} false & : & u(n) < MN || u(n) > MX \\ true & : & MN \le u(n) \le MX \end{array} \right. \tag{2.1}$$

2.3.1.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
[a,b]	1 MX: real 2 MN: real 3 u: real	1 y: logic		

2.3.1.3 Pseudo-code

System-Init-Code	Run-Code

	if ((MN <= u)&&(u <= MX)) {
	    y = 1;
	}
	else {
	    y = 0;
	}

2.3.1.4 Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6

Input values:
u 0 1 2 3 4 5 4
MN 1 1 1 1 4 1 1
MX 4 4 4 1 1 4 4

Output/state values:
y 0 1 1 0 0 0 1
```

Simulation result

Input/output relationship
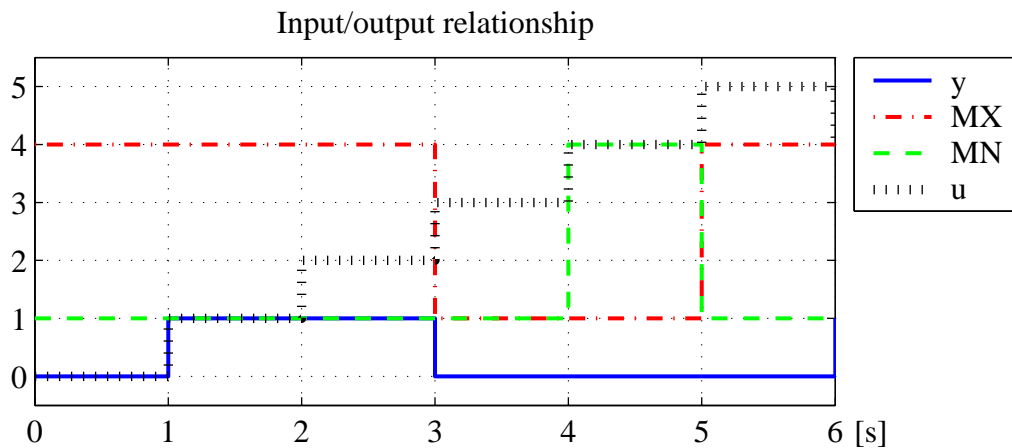
2.3.2   EQ (EQ)

2.3.2.1   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
=	1 u1: real 2 u2: real	1 y: logic		

2.3.2.2   Pseudo-code

System-Init-Code	Run-Code
	y = (u1 == u2);

2.3.3   GE (GE)

2.3.3.1   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
≥	1 u1: real 2 u2: real	1 y: logic		

2.3.3.2   Pseudo-code

System-Init-Code	Run-Code
	y = (u1 >= u2);

2.3.4 GT (GT)

2.3.4.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
>	1 u1: real 2 u2: real	1 y: logic		

2.3.4.2 Pseudo-code

System-Init-Code	Run-Code
	y = (u1 > u2);

2.3.5 LE (LE)

2.3.5.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
≤	1 u1: real 2 u2: real	1 y: logic		

2.3.5.2 Pseudo-code

System-Init-Code	Run-Code
	y = (u1 <= u2);

2.3.6 LT (LT)

2.3.6.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
<	1 u1: real 2 u2: real	1 y: logic		

2.3.6.2 Pseudo-code

System-Init-Code	Run-Code
	y = (u1 < u2);

2.3.7   LeftOpenInterval (LEFTOPENINTERVAL)

2.3.7.1   Verbal description

$$y(n) = \begin{cases} false & : & u(n) <= MN || u(n) > MX \\ true & : & MN < u(n) \leq MX \end{cases} \tag{2.2}$$

2.3.7.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
]a,b]	1 MX: real 2 MN: real 3 u: real	1 y: logic		

2.3.7.3   Pseudo-code

System-Init-Code	Run-Code

	if ((MN < u)&&(u <= MX)) {
	        y = 1;
	}
	else {
	        y = 0;
	}

2.3.7.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6

Input values:
u 0 1 2 3 4 5 4
MN 1 1 1 1 4 1 1
MX 4 4 4 1 1 4 4

Output/state values:
y 0 0 1 0 0 0 1
```

Simulation result

Input/output relationship
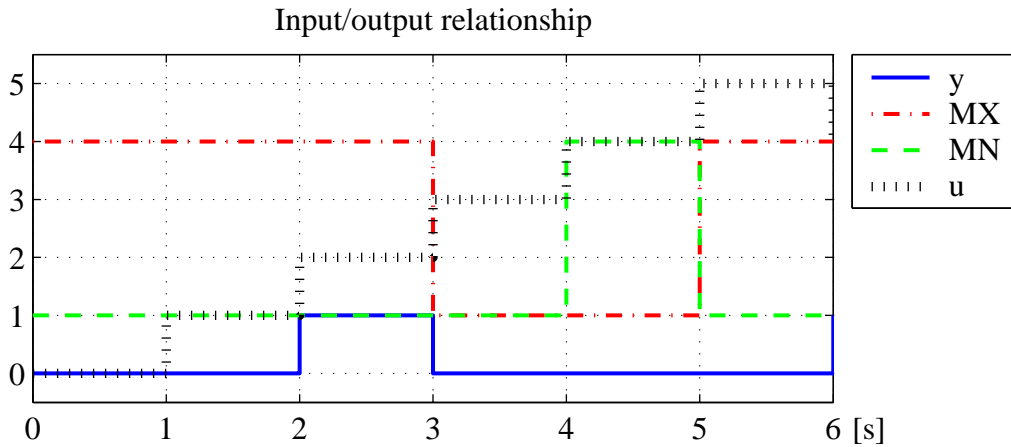
2.3.8   NE (NEQ)

2.3.8.1   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
$\neq$	1 u1: real 2 u2: real	1 y: logic		

2.3.8.2   Pseudo-code

System-Init-Code	Run-Code
	y = (u1 != u2);

2.3.9   OpenInterval (OPENINTERVAL)

2.3.9.1   Verbal description

$$y(n) = \begin{cases} false & : & u(n) \leq MN || u(n) \geq MX \\ true & : & MN < u(n) < MX \end{cases} \tag{2.3}$$

2.3.9.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
]a,b[	1 MX: real 2 MN: real 3 u: real	1 y: logic		

2.3.9.3   Pseudo-code

System-Init-Code	Run-Code

	if ((MN < u)&&(u < MX)) {
	    y = 1;
	}
	else {
	    y = 0;
	}

2.3.9.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6

Input values:
u 0 1 2 3 4 5 4
MN 1 1 1 1 4 1 1
MX 4 4 4 1 1 4 4

Output/state values:
y 0 0 1 0 0 0 0
```

Simulation result

Input/output relationship
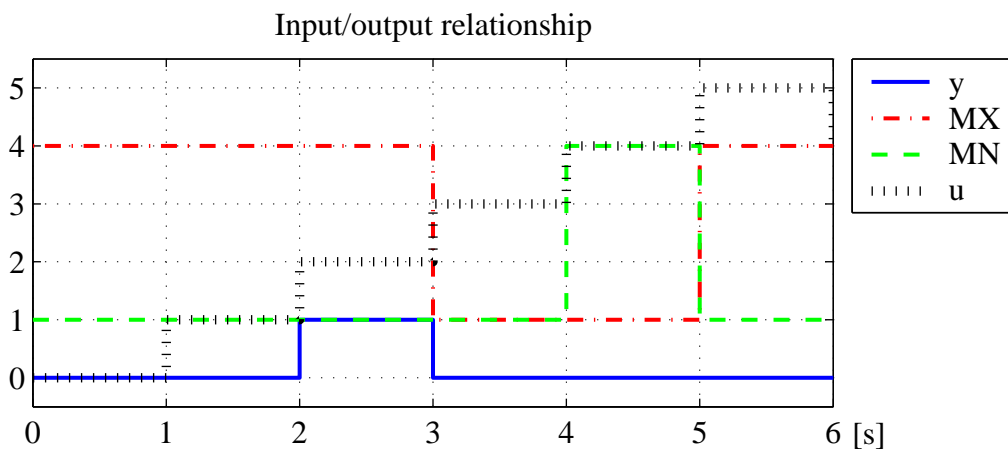
2.3.10 RightOpenInterval (RIGHTOPENINTERVAL)

2.3.10.1 Verbal description

$$y(n) = \begin{cases} false & : & u(n) < MN || u(n) \geq MX \\ true & : & MN \leq u(n) < MX \end{cases} \tag{2.4}$$

2.3.10.2 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
[a,b[	1 MX: real 2 MN: real 3 u: real	1 y: logic		

2.3.10.3 Pseudo-code

System-Init-Code	Run-Code

	if ((MN <= u)&&(u < MX)) {
	        y = 1;
	}
	else {
	        y = 0;
	}

2.3.10.4 Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6

Input values:
u 0 1 2 3 4 5 4
MN 1 1 1 1 4 1 1
MX 4 4 4 1 1 4 4

Output/state values:
y 0 1 1 0 0 0 0
```

Simulation result

Input/output relationship
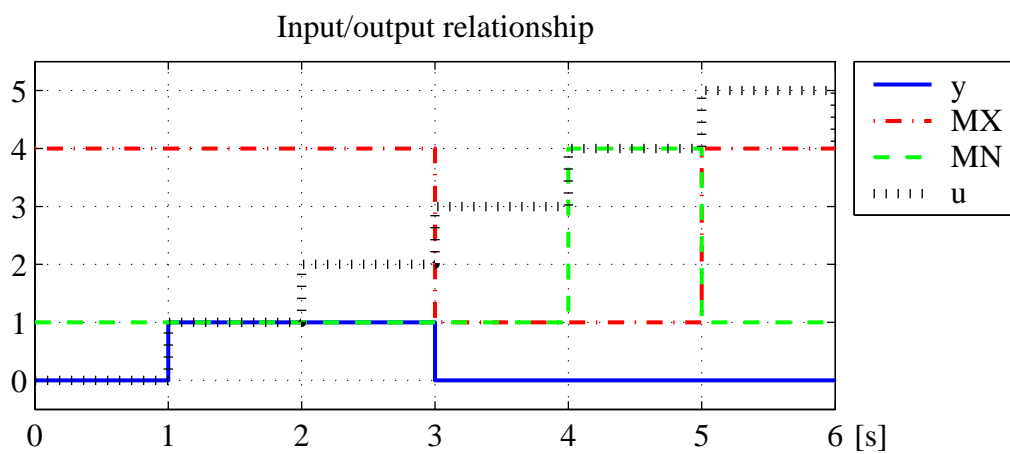
2.4 Mathematical functions

2.4.1 InverseCosine (ACOS)

2.4.1.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
ACOS	1 u: real	1 y: real		

2.4.1.2 Pseudo-code

System-Init-Code	Run-Code
	y = acos(u);

2.4.2 InverseSine (ASIN)

2.4.2.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
ASIN	1 u: real	1 y: real		

2.4.2.2 Pseudo-code

System-Init-Code	Run-Code
	y = asin(u);

2.4.3 InverseTangent (ATAN)

2.4.3.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
ATAN	1 u: real	1 y: real		

2.4.3.2 Pseudo-code

System-Init-Code	Run-Code
	y = atan(u);

2.4.4 Cosine (COS)

2.4.4.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
COS	1 u: real	1 y: real		

2.4.4.2 Pseudo-code

System-Init-Code	Run-Code
	y = cos(u);

2.4.5 Exponential (EXP)

2.4.5.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
EXP	1 u: real	1 y: real		

2.4.5.2 Pseudo-code

System-Init-Code	Run-Code
	y = exp(u);

2.4.6 NaturalLogarithm (LOG)

2.4.6.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
LOG	1 u: real	1 y: real		

2.4.6.2 Pseudo-code

System-Init-Code	Run-Code
	y = log(u);

2.4.7 Power (POW)

2.4.7.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
POW	1 u1: real 2 u2: real	1 y: real		

2.4.7.2 Pseudo-code

System-Init-Code	Run-Code
	y = pow(u1,u2);

2.4.8 Sine (SIN)

2.4.8.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
SIN	1 u: real	1 y: real		

2.4.8.2 Pseudo-code

System-Init-Code	Run-Code
	y = sin(u);

2.4.9 Square (SQR)

2.4.9.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
SQR	1 u: real	1 y: real		

2.4.9.2 Pseudo-code

System-Init-Code	Run-Code
	y = pow(u,2);

2.4.10 SquareRoot (SQRT)

2.4.10.1 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
SQRT	1 u: real	1 y: real		

2.4.10.2 Pseudo-code

System-Init-Code	Run-Code
	y = sqrt(u);

2.4.11   Tangent (TAN)

2.4.11.1   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
TAN	1 u: real	1 y: real		

2.4.11.2   Pseudo-code

System-Init-Code	Run-Code
	y = tan(u);

2.5 Counter and timer

Concerning the reset functionality please refer to the description at list entry 5 on page 9. The rounding function used in this section is described at list entry 4 on page 8.

2.5.1 CountDownResetEnabled (COUNTDOWN_RE)

2.5.1.1 Verbal description

Output is true, if the number of block evaluations since the last reset is less than the initial number of block evaluations IV.

2.5.1.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 E: logic 2 R: logic 3 IV: real	1 y: logic	1 x: int	

2.5.1.3 Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	        x = round(IV);
	}
	else if (E) {
	        if (x > 0) {
	                x = x - 1;
	        }
	}
	y = (x > 0);

2.5.1.4   Test-cases

Testdata
Time vector:
```
t 0 1 2 3 4 5 6
```

Input values:
```
E 1 0 1 0 1 1 1
R 1 1 1 0 0 0 0
IV 0.5 0.5 1.5 3 3 3 3
```

Output/state values:
```
x 1 1 2 2 1 0 0
y 1 1 1 1 1 0 0
```

Simulation result
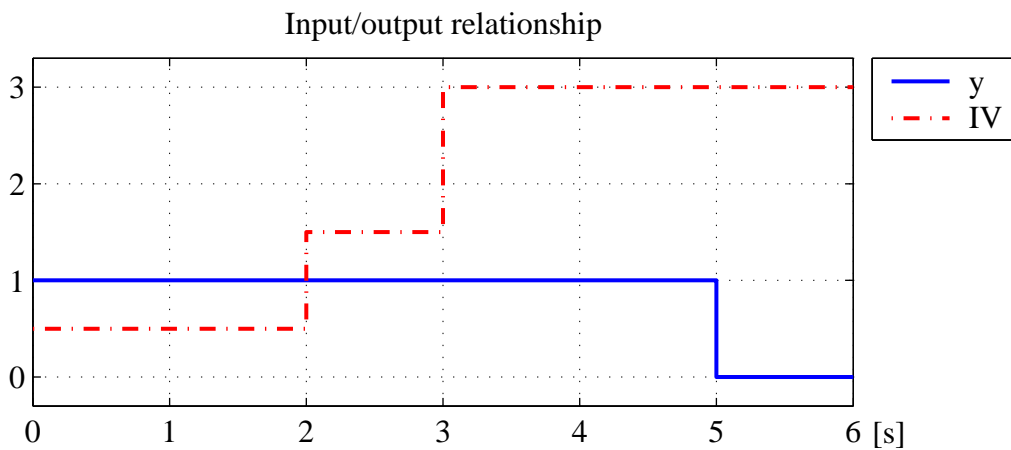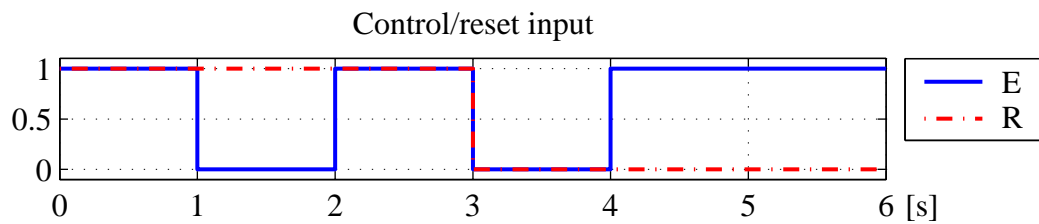
2.5.2   CountDownResetTriggerEnabled (COUNTDOWN_RTE)

2.5.2.1   Verbal description

Output is true, if the number of block evaluations since the last reset is less than the initial number of block evaluations IV.

2.5.2.2   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 E: logic 2 RT: logic 3 IV: real	1 y: logic	1 x1: int 2 x2: logic	

2.5.2.3   Pseudo-code

System-Init-Code	Run-Code

x1 = 0;
x2 = 0;
	if (RT && !x2) {
	    x1 = round(IV);
	}
	else if (E) {
	    if (x1 > 0) {
	        x1 = x1 - 1;
	    }
	}
	x2 = RT;
	y = (x1 > 0);

2.5.2.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5

Input values:
E 0 1 1 1 1 1
RT 1 1 0 1 1 1
IV 0.5 0.5 1.5 1.5 3 3

Output/state values:
x1 1 0 0 2 1 0
x2 1 1 0 1 1 1
y 1 0 0 1 1 0
```

Simulation result

Control/reset input

Input/output relationship
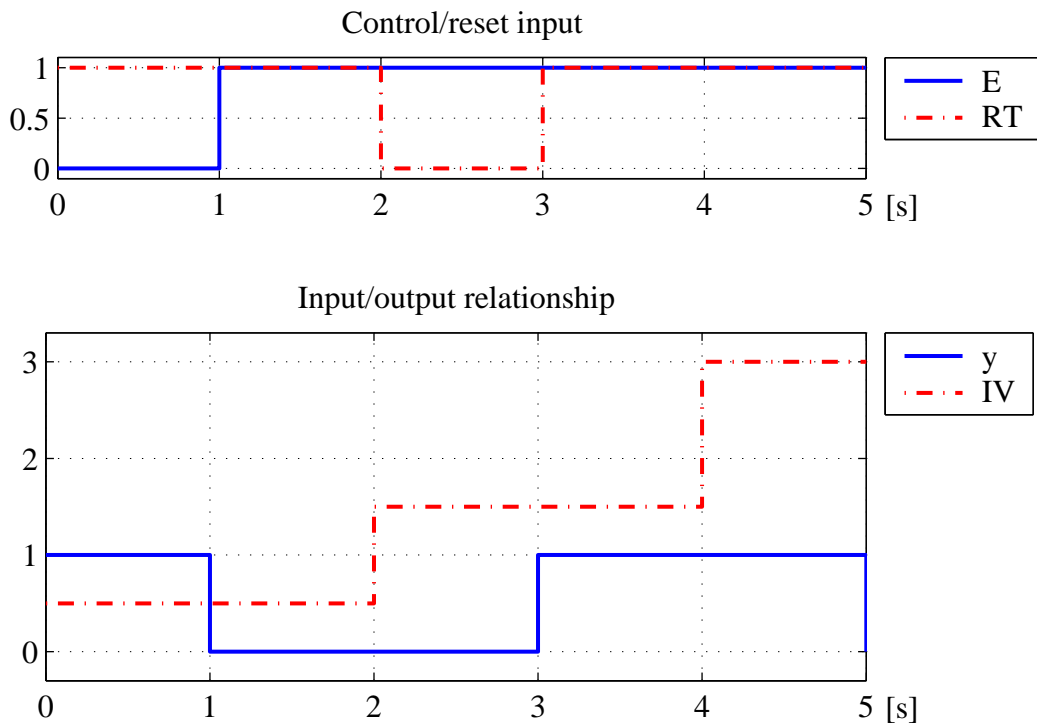
2.5.3   CounterResetEnabled (COUNTER_RE)

2.5.3.1   Verbal description

Counts up and outputs the number of block evaluations since the last reset.

2.5.3.2   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 E: logic 2 R: logic 3 IV: real	1 y: real	1 x: int	

2.5.3.3   Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	    x = round(IV);
	}
	else if (E) {
	    x = x + 1;
	}
	y = x;

2.5.3.4 Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4

Input values:
E 1 1 1 1 0
R 0 0 1 0 0
IV 1 1 1 1 1

Output/state values:
x 1 2 1 2 2
y 1 2 1 2 2
```

Simulation result
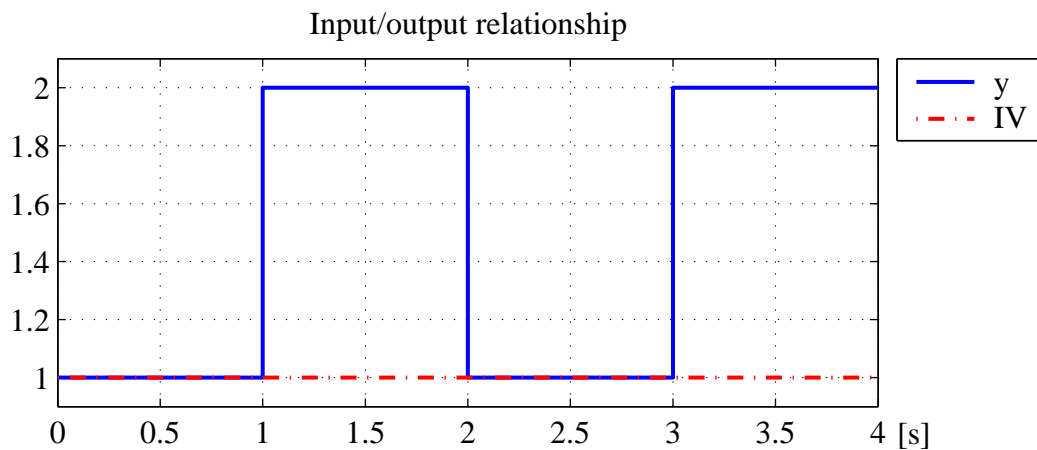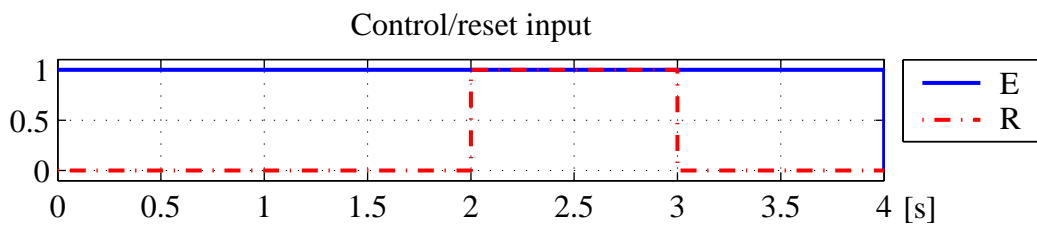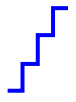
2.5.4 CounterResetTriggerEnabled (COUNTER_RTE)

2.5.4.1 Verbal description

Counts up and outputs the number of block evaluations since the last reset.

2.5.4.2   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 E: logic 2 RT: logic 3 IV: real	1 y: real	1 x1: real 2 x2: logic	

2.5.4.3   Pseudo-code

System-Init-Code	Run-Code

x1 = 0;
x2 = 0;
	if (RT && !x2) {
	    x1 = round(IV);
	}
	else if (E) {
	    x1 = x1 + 1;
	}
	x2 = RT;
	y = x1;

2.5.4.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6

Input values:
E 1 1 0 1 0 1 1
RT 0 0 0 1 1 1 0
IV 1 1 1 1 1 1 1

Output/state values:
x1 1 2 2 1 1 2 3
x2 0 0 0 1 1 1 0
y 1 2 2 1 1 2 3
```

Simulation result

Input/output relationship
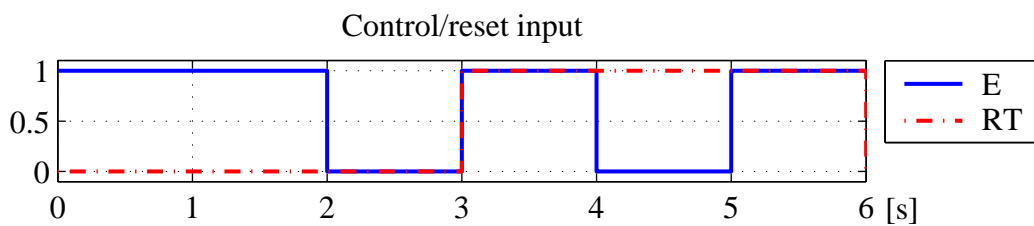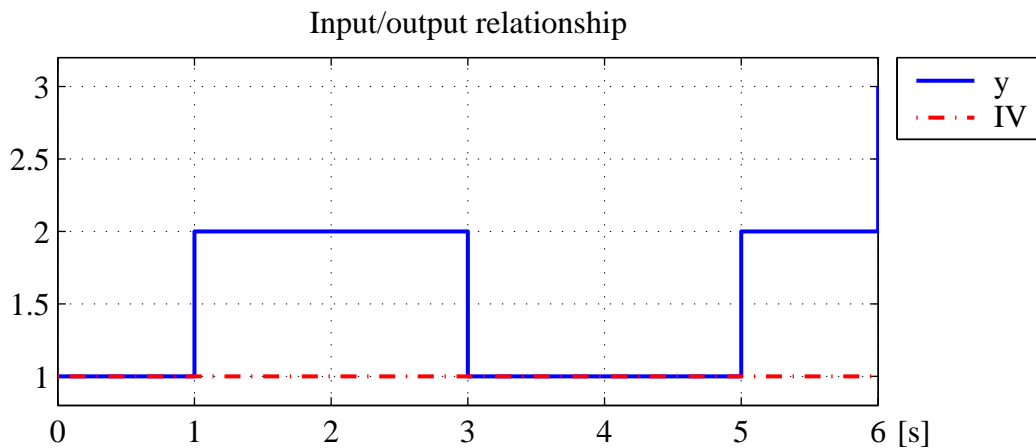
2.5.5 StopWatchResetEnabled (STOPWATCH_RE)

2.5.5.1 Verbal description

The block outputs the time as a multiple of the sample time dT since the system init or the last reset.

2.5.5.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 E: logic 2 R: logic	1 y: real	1 x: real	

2.5.5.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    x = 0.0;
	}
	else if (E) {
	    x = x + dT;
	}
	y = x;

2.5.5.4 Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6 7

Input values:
E 1 1 1 1 1 0 0 0
R 0 0 1 0 0 0 0 1

Output/state values:
x 1 2 0 1 2 2 2 0
y 1 2 0 1 2 2 2 0
```

Simulation result
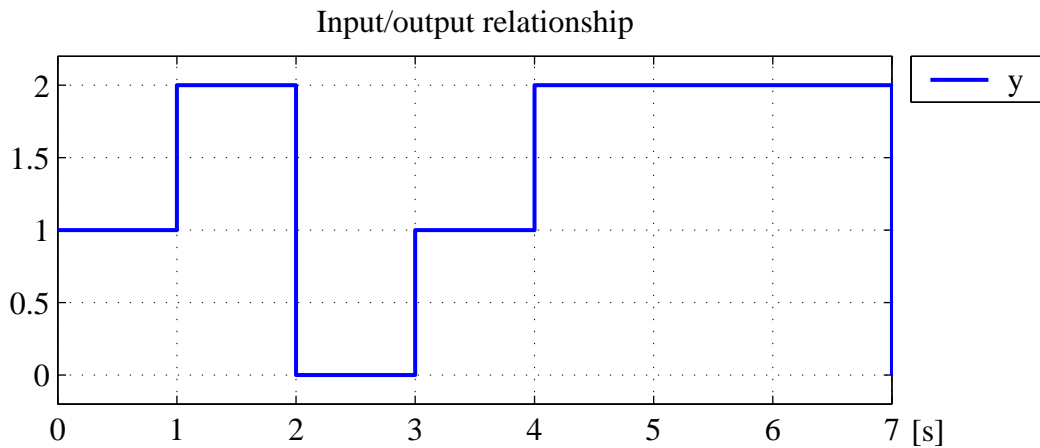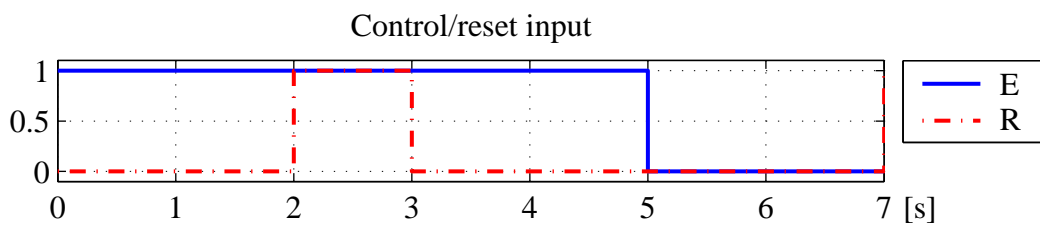
2.5.6 StopWatchResetTriggerEnabled (STOPWATCH_RTE)

2.5.6.1 Verbal description

The block outputs the time as multiple of the sample time dT since the system init or the last reset.

2.5.6.2   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 E: logic 2 RT: logic	1 y: real	1 x1: real 2 x2: logic	

2.5.6.3   Pseudo-code

System-Init-Code	Run-Code

x1 = 0.0;
x2 = 0;
	if (RT && !x2) {
	     x1 = 0.0;
	}
	else if (E) {
	     x1 = x1 + dT;
	}
	x2 = RT;
	y = x1;

2.5.6.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6

Input values:
E 1 1 0 1 0 1 1
RT 0 0 0 1 1 1 0

Output/state values:
x1 1 2 2 0 0 1 2
x2 0 0 0 1 1 1 0
y 1 2 2 0 0 1 2
```

Simulation result

Input/output relationship
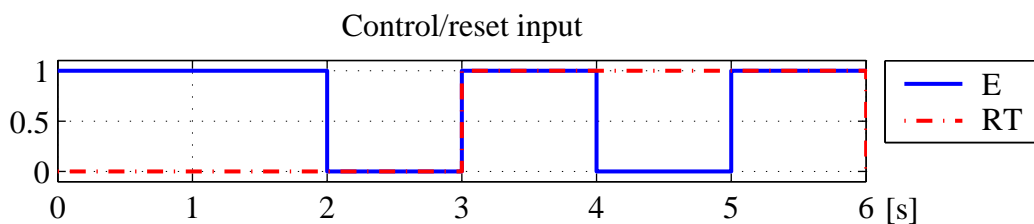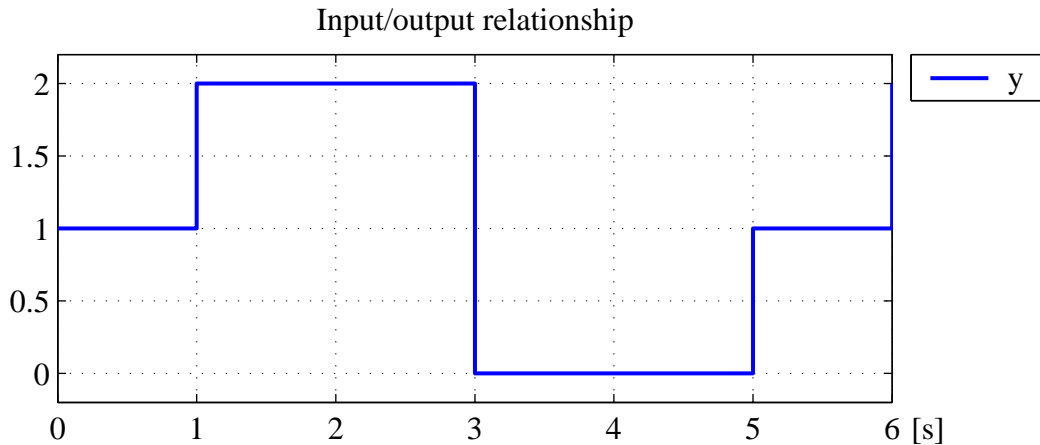
2.5.7   TimerRetriggerResetEnabled (TIMERRETRIGGER_RE)

2.5.7.1   Verbal description

The block indicates true as long as the number of block evaluations since the *last reset* is smaller than the initial time *IV* set at the reset divided by the sample time *dT*. Assuming the reset at time $t = i \cdot dT$, i.e. $R(i) = true$, the block can be represented as:

$$y(i + k) = \begin{cases} false & : & k \leq \frac{IV(i)}{dT} \\ true & : & k > \frac{IV(i)}{dT} \end{cases} \qquad (2.5)$$

A triggered reset is always possible and independent of the current state..

2.5.7.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 E: logic 2 R: logic 3 IV: real	1 y: logic	1 x: real	

2.5.7.3   Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    x = IV;
	}
	else if (E) {
	    x = MAX(x - dT, 0.0);
	}
	y = (x > 0.0);

2.5.7.4 Test-cases

Testdata
Time vector:
t 0 1 2 3 4 5 6

Input values:
E 1 0 1 0 1 1 1
R 1 1 0 0 0 0 0
IV -3 3 3 3 3 3 3

Output/state values:
x -3 3 2 2 1 0 0
y 0 1 1 1 1 0 0

Simulation result

Control/reset input

Input/output relationship
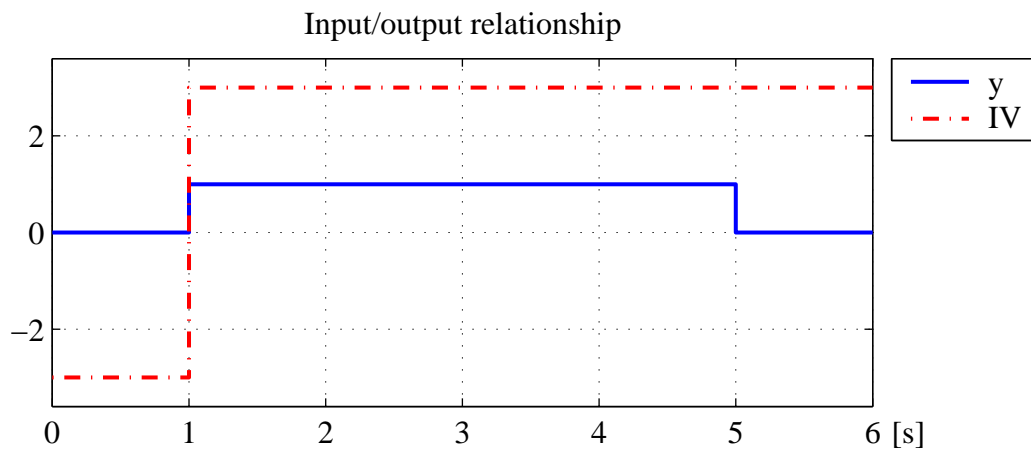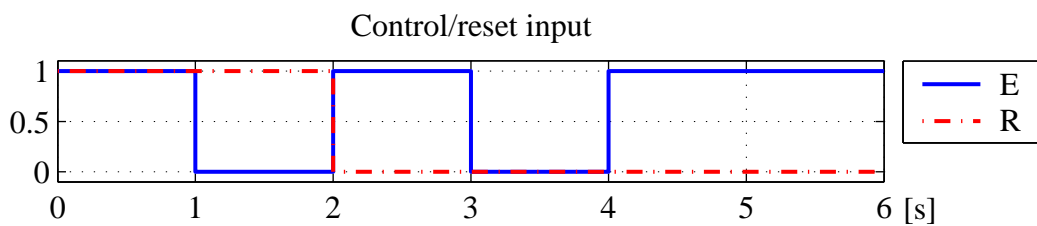
2.5.8 TimerRetriggerResetTriggerEnabled (TIMERRETRIGGER_RTE)

2.5.8.1 Verbal description

Please refer to equation 2.5 on page 36 for the verbal description.

2.5.8.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 E: logic 2 RT: logic 3 IV: real	1 y: logic	1 x1: real 2 x2: logic	

2.5.8.3   Pseudo-code

System-Init-Code	Run-Code

x1 = 0.0;
x2 = 0;
	if (RT && !x2) {
	    x1 = IV;
	}
	else if (E) {
	    x1 = MAX(x1 - dT, 0.0);
	}
	x2 = RT;
	y = (x1 > 0.0);

2.5.8.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6

Input values:
E 1 1 0 1 1 1 1
RT 1 0 1 1 1 0 0
IV -3 3 3 3 3 3 3

Output/state values:
x1 -3 0 3 2 1 0 0
x2 1 0 1 1 1 0 0
y 0 0 1 1 1 0 0
```

Simulation result

Control/reset input

Input/output relationship
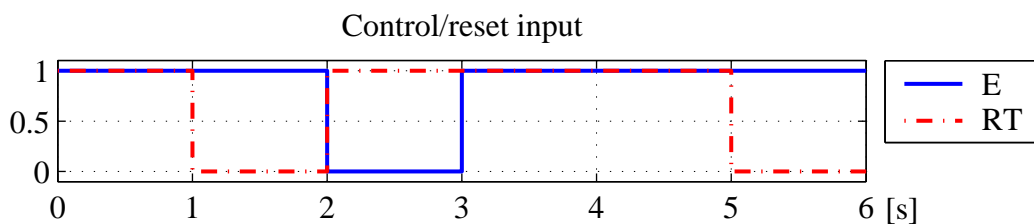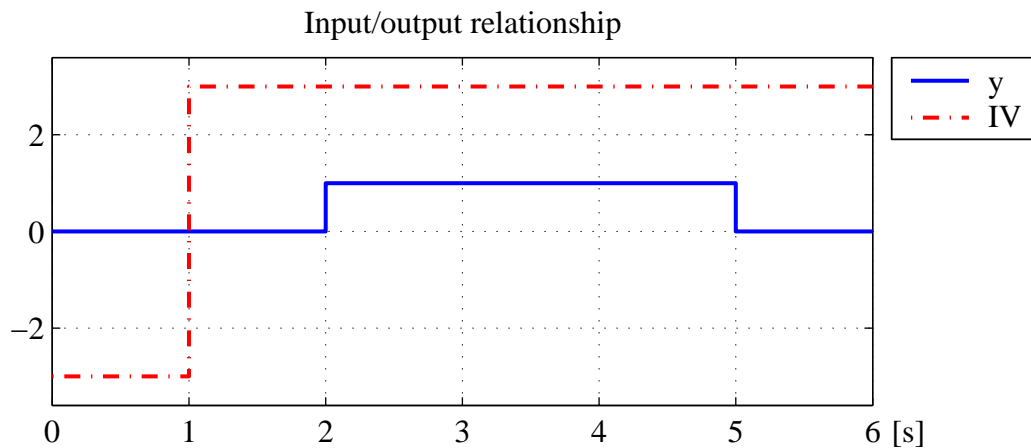
2.5.9   TimerResetEnabled (TIMER_RE)

2.5.9.1   Verbal description

The block indicates true, if the initial time IV set at the last reset has not yet run down to zero. Reset can only be performed, after the counter has run down to zero.

2.5.9.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 E: logic 2 R: logic 3 IV: real	1 y: logic	1 x: real	

2.5.9.3   Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    if (x <= 0.0) {
	        x = IV;
	    }
	}
	else if (E) {
	    x = MAX(x - dT, 0.0);
	}
	y = (x > 0.0);

2.5.9.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6 7

Input values:
E 1 0 0 0 1 1 1 1
R 1 1 1 0 0 1 0 0
IV -3 3 -3 3 3 3 3 3

Output/state values:
x -3 3 3 3 2 2 1 0
y 0 1 1 1 1 1 1 0
```

Simulation result

Control/reset input

Input/output relationship
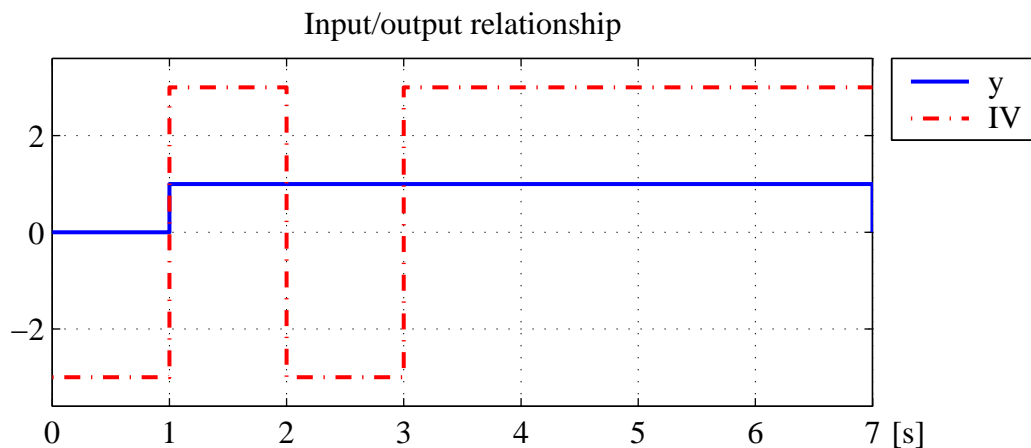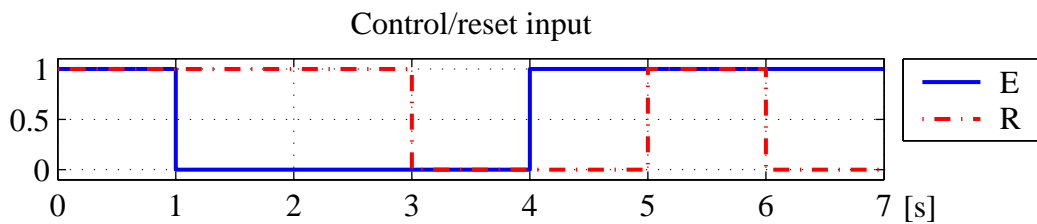
2.5.10   TimerResetTriggerEnabled (TIMER_RTE)

2.5.10.1   Verbal description

The block indicates true, if the initial time IV set at the last reset has not yet run down to zero. Reset can only be performed, after the counter has run down to zero.

2.5.10.2 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 E: logic 2 RT: logic 3 IV: real	1 y: logic	1 x1: real 2 x2: logic	

2.5.10.3 Pseudo-code

System-Init-Code	Run-Code

x1 = 0.0;
x2 = 0;
	if (RT && !x2) {
	    if (x1 <= 0.0) {
	        x1 = IV;
	    }
	}
	else if (E) {
	    x1 = MAX(x1 - dT, 0.0);
	}
	x2 = RT;
	y = (x1 > 0.0);

2.5.10.4 Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7

Input values:
E  1 1 0 1 0 1 1 1
RT 1 0 1 0 1 0 0 0
IV -3 3 3 3 3 3 3 3

Output/state values:
x1 -3 0 3 2 2 1 0 0
x2  1 0 1 0 1 0 0 0
y   0 0 1 1 1 1 0 0
```

Simulation result

Control/reset input

Input/output relationship
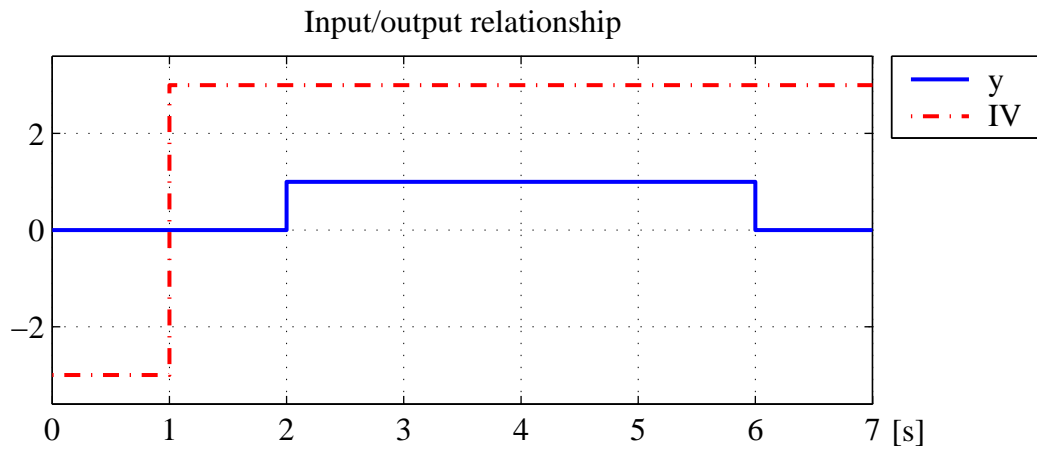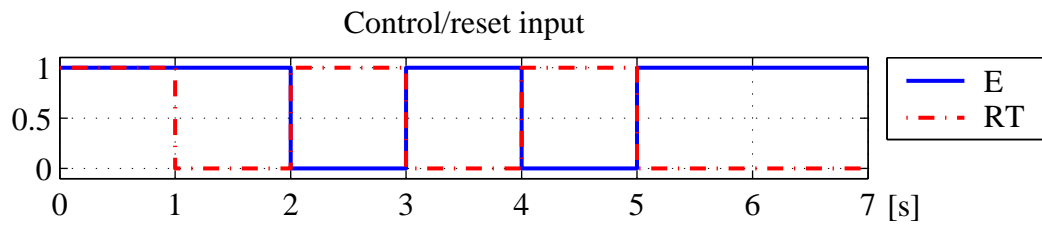
2.6 Delay blocks

2.6.1 DelayResetEnabled (DELAY_RE)

2.6.1.1 Verbal description

Delays the logical or real input signal u by one sample time dT. The reset value IV has direct influence on the output.

2.6.1.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
$\mathbf{z^{-1}}$	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real, logic	1 x: real	

2.6.1.3 Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	        y = IV;
	        x = IV;
	}
	else if (E) {
	        y = x;
	        x = u;
	}
	else {
	        y = x;
	}

2.6.1.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5

Input values:
u 0 1 2 3 4 5
E 0 1 1 1 0 0
R 1 1 0 0 0 0
IV 3 -3 3 3 3 3

Output/state values:
x 3 -3 2 3 3 3
y 3 -3 -3 2 3 3
```

Simulation result

Control/reset input

Input/output relationship
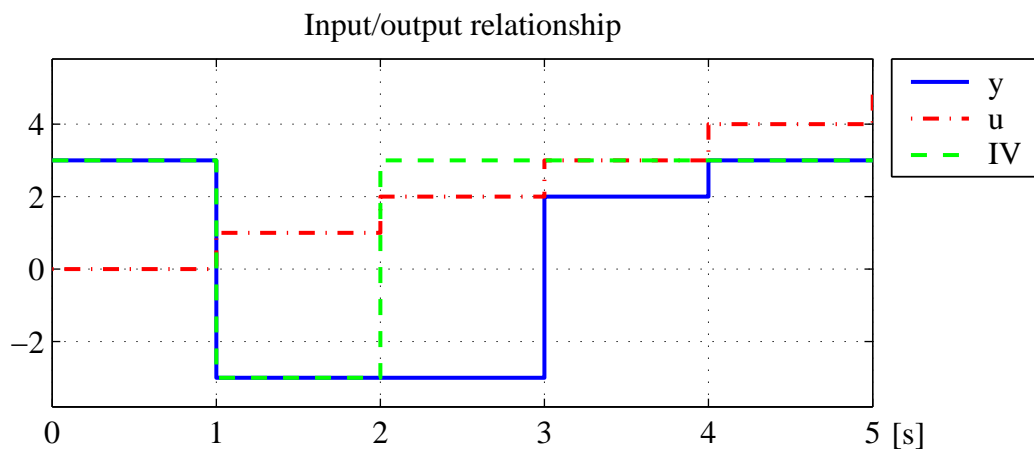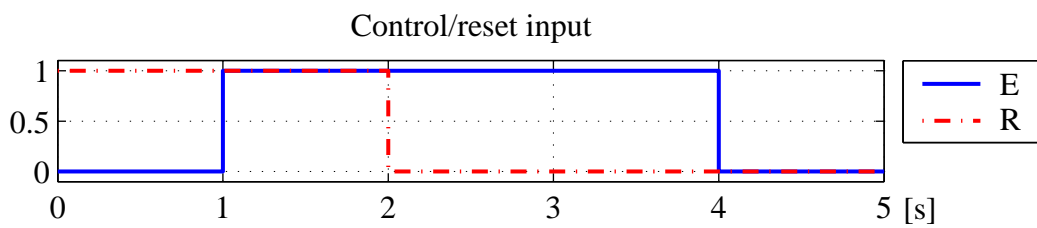
2.6.2 TurnOffDelaySample (TURNOFFDELAYSAMPLE)

2.6.2.1 Verbal description

A falling edge at time $t = i \cdot dT$ of the input signal u is delayed by n block evaluations.

$$y(i + k) = \begin{cases} false & : & u(i + k) == false \&\& k \leq n \\ true & : & u(i + k) == true || k > n; 0(1)k \end{cases} \tag{2.6}$$

2.6.2.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 n: real 2 u: logic 3 R: logic	1 y: logic	1 x: int	1 temp: int

2.6.2.3 Pseudo-code

System-Init-Code	Run-Code

x = 0;	```
	if (R) {
	    x = round(n);
	}
	temp = x;
	if (u) {
	    x = round(n);
	}
	else {
	    if (temp > 0) {
	        x = x - 1;
	    }
	}
	y = ((temp > 0) || u);
	``

2.6.2.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
n -1.5 -1.5 -1.5 2 2 2 2 2 2 2 2
u 1 0 0 1 1 0 0 0 0 0 0
R 0 0 1 0 0 0 0 1 0 0 0

Output/state values:
x -1 -1 -1 2 2 1 0 1 0 0 0
y 1 0 0 1 1 1 1 1 1 0 0
```

Simulation result

Parameter input

Control/reset input

Input/output relationship
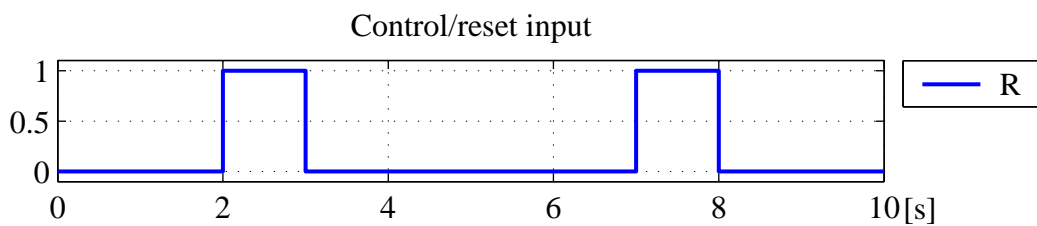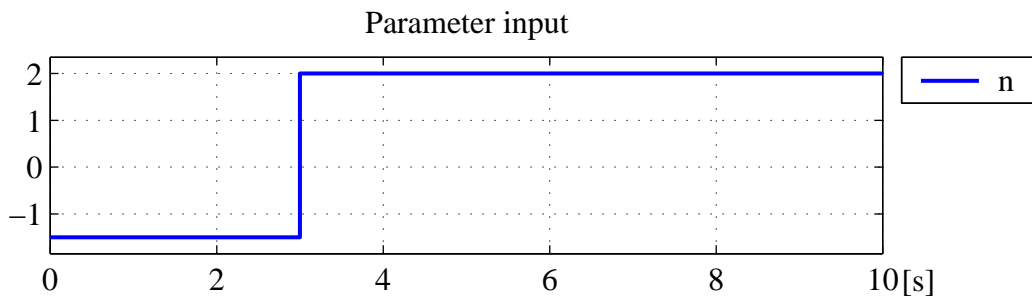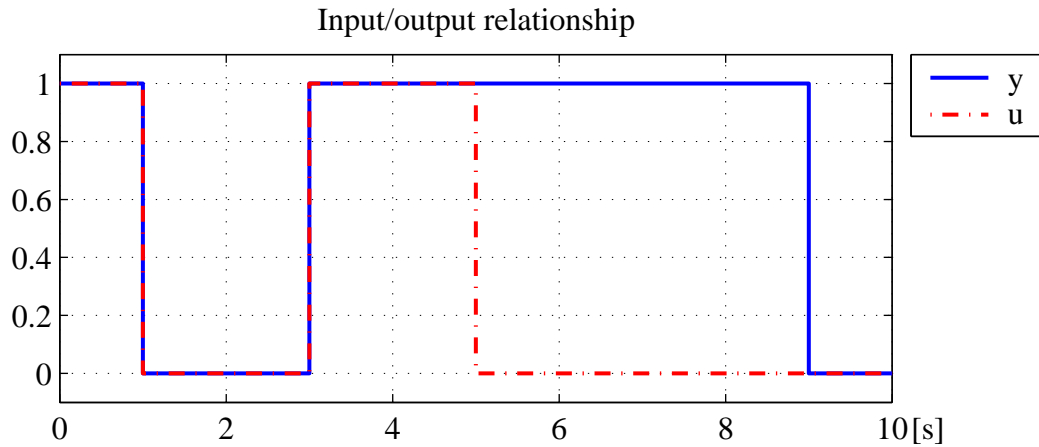
2.6.3 TurnOffDelayTime (TURNOFFDELAYTIME)

2.6.3.1 Verbal description

A falling edge of the input signal u is delayed by the time T, if the input remains false for minimum the whole period T. The relationship between the time delay t and the number of delay samples n is

$$n = \left\lceil \frac{T}{dT} \right\rceil \tag{2.7}$$

2.6.3.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
T	1 T: real 2 u: logic 3 R: logic	1 y: logic	1 x: real	1 temp: real

2.6.3.3 Pseudo-code

System-Init-Code	Run-Code
x = 0.0;	```if (R) { x = T; } temp = x; if (u) { x = T; } else { if (temp > 0.0) { x = x - dT; } } y = ((temp > 0.0) || u);```

2.6.3.4 Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
T -2 -2 -2 2 2 2 2 2 2 2 2
u 1 0 0 1 1 0 0 0 0 0 0
R 0 0 1 0 0 0 0 1 0 0 0

Output/state values:
x -2 -2 -2 2 2 1 0 1 0 0 0
y 1 0 0 1 1 1 1 1 1 0 0
```
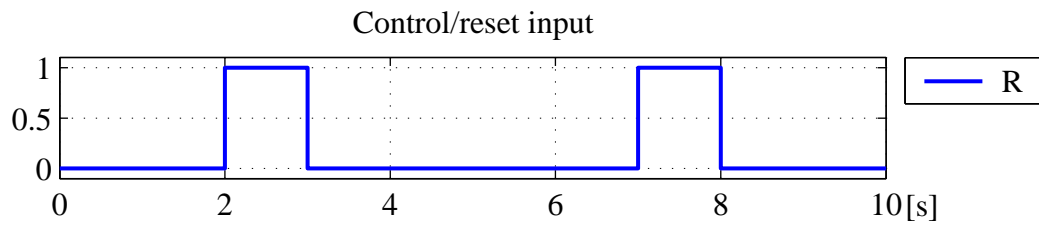
Simulation result

Parameter input

Control/reset input

Input/output relationship
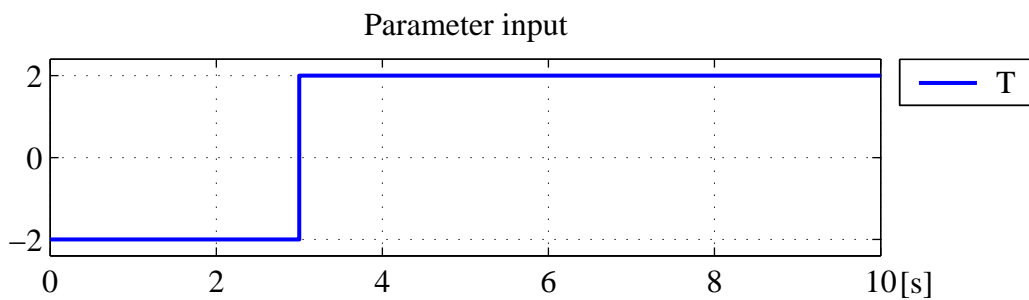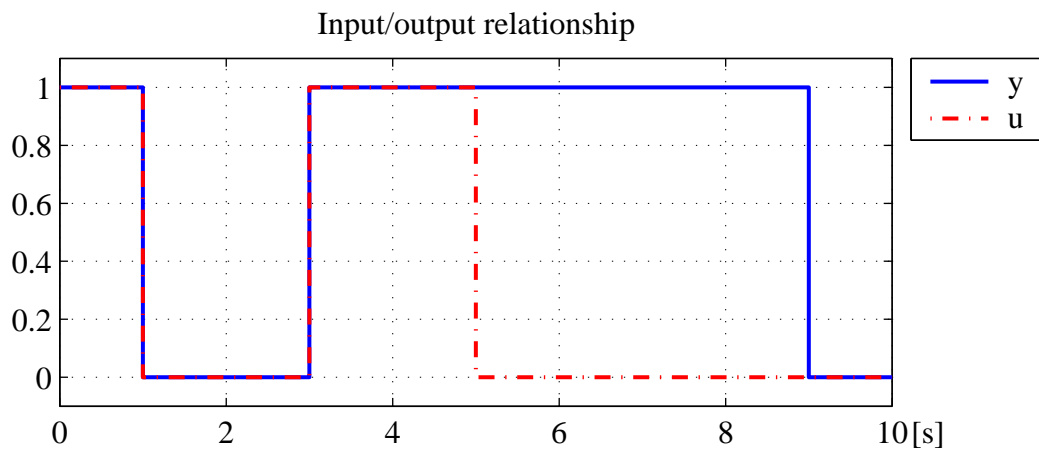
2.6.4 TurnOnDelaySample (TURNONDELAYSAMPLE)

2.6.4.1 Verbal description

A rising edge of the input signal u is delayed by n block evaluations (samples).

$$y(i + k) = \begin{cases} true & : & u(i + k) == true \&\&k \leq n \\ false & : & u(i + k) == false || k > n \end{cases} \tag{2.8}$$

2.6.4.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
$\mathbf{n}$	1 n: real 2 u: logic 3 R: logic	1 y: logic	1 x: int	1 temp: int

2.6.4.3   Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	        x = round(n);
	}
	temp = x;
	if (u) {
	        if (temp > 0) {
	                x = x - 1;
	        }
	}
	else {
	        x = round(n);
	}
	y = ((temp <= 0) && u);

2.6.4.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
n -1.5 -1.5 -1.5 2 2 2 2 2 2 2 2
u 0 1 1 0 0 1 1 1 1 1 1 1
R 0 0 1 0 0 0 0 1 0 0 0

Output/state values:
x -1 -1 -1 2 2 1 0 1 0 0 0
y 0 1 1 0 0 0 0 0 0 1 1
```
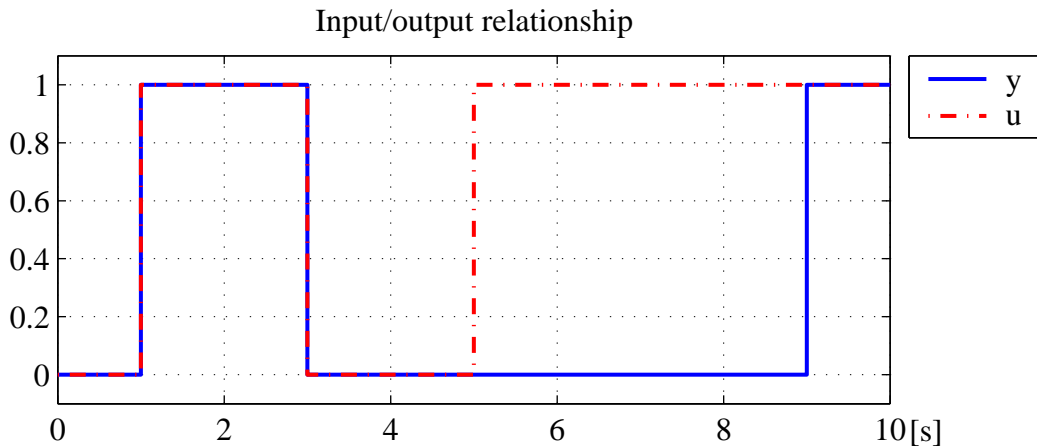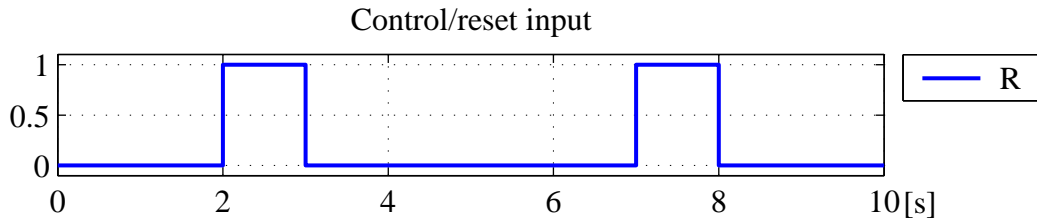
Simulation result

Parameter input

Control/reset input

Input/output relationship
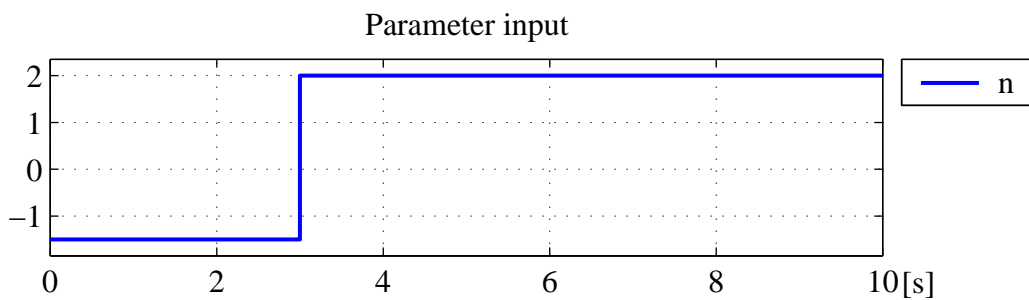
2.6.5 TurnOnDelayTime (TURNONDELAYTIME)

2.6.5.1 Verbal description

A rising edge of the input signal u is delayed by the time T, if the input signal remains high (true) for minimum this period T. The relationship between the time delay t and the number of delay samples n is

$$n = \left\lceil \frac{T}{dT} \right\rceil \tag{2.9}$$

2.6.5.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 T: real 2 u: logic 3 R: logic	1 y: logic	1 x: real	1 temp: real

2.6.5.3   Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    x = T;
	}
	temp = x;
	if (u) {
	    if (temp > 0.0) {
	        x = x - dT;
	    }
	}
	else {
	    x = T;
	}
	y = ((temp <= 0.0) && u);

2.6.5.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
T -2 -2 -2 2 2 2 2 2 2 2 2
u 0 1 1 0 0 1 1 1 1 1 1
R 0 0 1 0 0 0 0 1 0 0 0

Output/state values:
x -2 -2 -2 2 2 1 0 1 0 0 0
y 0 1 1 0 0 0 0 0 0 1 1
```
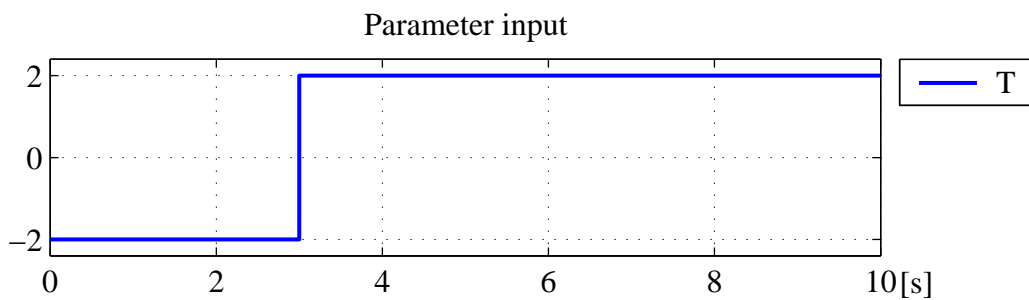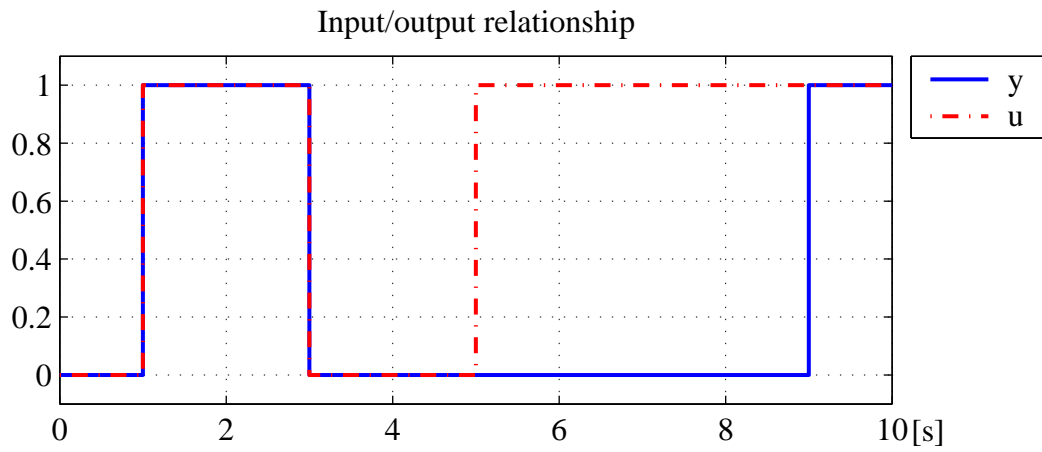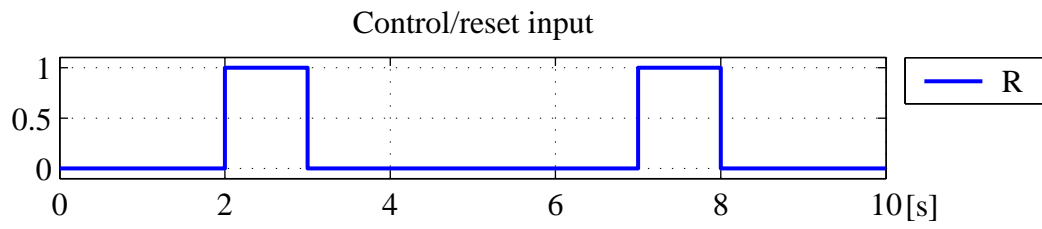
Simulation result

Parameter input

Control/reset input

Input/output relationship

2.7 Memory blocks

2.7.1 DeltaOneStep (DELTAONESTEP)

2.7.1.1 Verbal description

Returns the difference between the current and the last input value.

2.7.1.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 u: real 2 R: logic 3 IV: real	1 y: real	1 x: real	

2.7.1.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	     x = IV;
	 }
	 y = u - x;
	 x = u;

2.7.1.4 Test-cases

Testdata
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
u 1 2 4 7 11 12 14 17 20
R 0 0 0 0 0 1 1 0 0
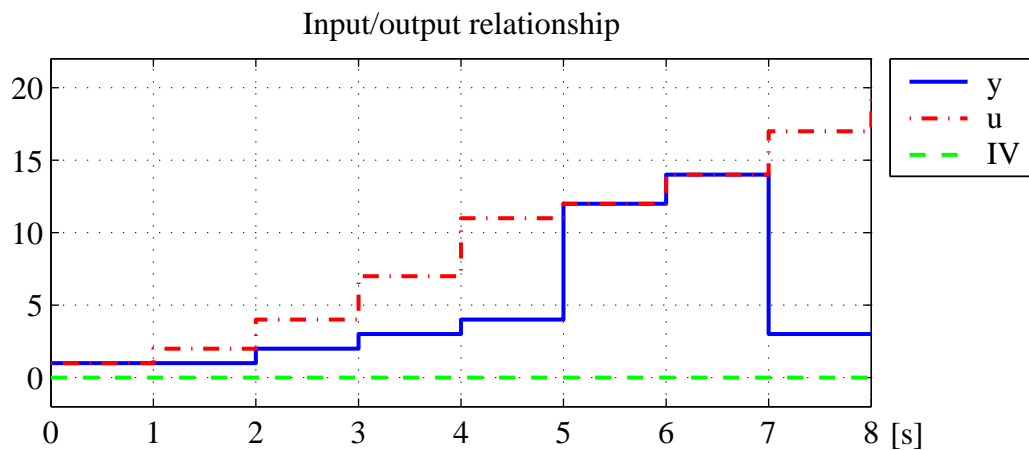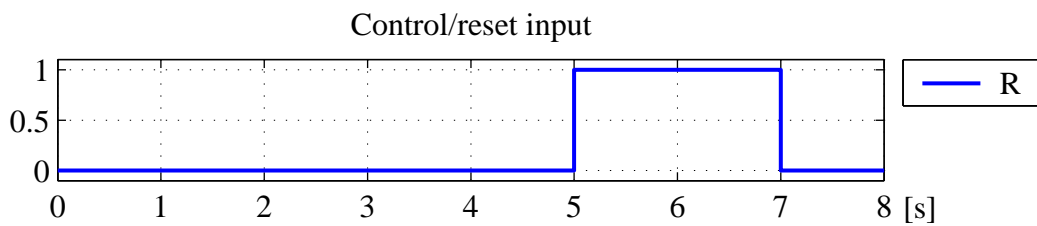IV 0 0 0 0 0 0 0 0 0

Output/state values:
x 1 2 4 7 11 12 14 17 20
y 1 1 2 3 4 12 14 3 3

Simulation result

Control/reset input

Input/output relationship
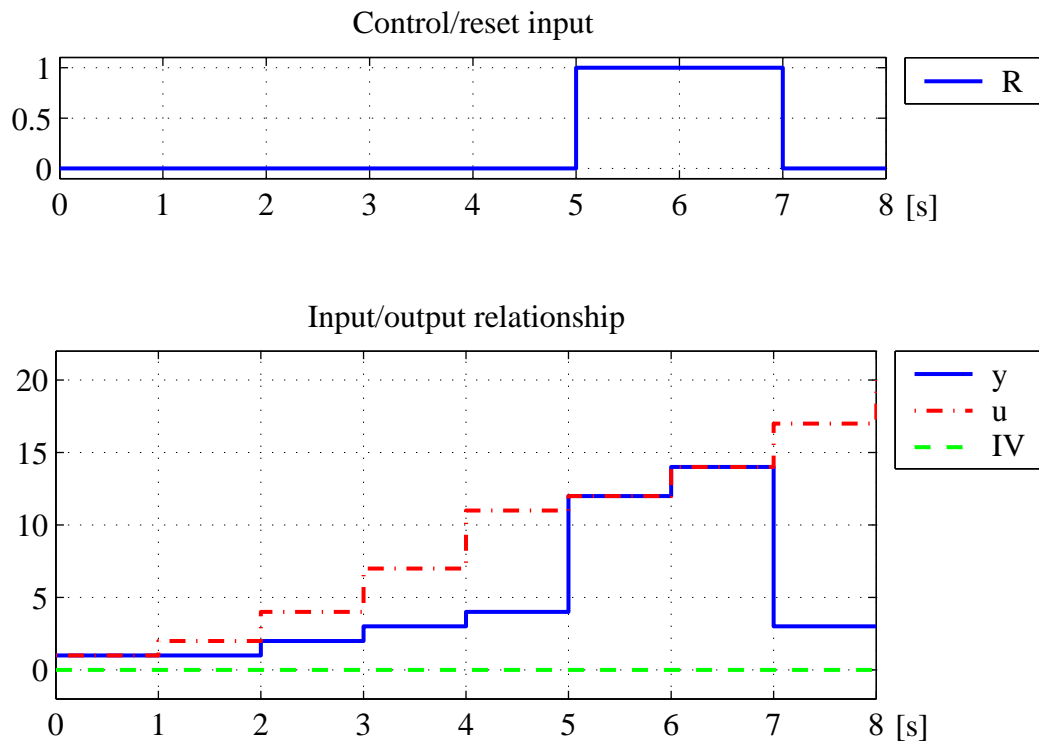
2.7.2 DifferenceQuotient (DIFFERENCEQUOTIENT)

2.7.2.1 Verbal description

Returns the rate of change of the input signal u over time $\frac{u(n) - u(n-1)}{dT}$.

2.7.2.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
$\Delta/\mathbf{dT}$	1 u: real 2 R: logic 3 IV: real	1 y: real	1 x: real	

2.7.2.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    x = IV;
	}
	y = (u - x)/dT;
	x = u;

2.7.2.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
u 1 2 4 7 11 12 14 17 20
R 0 0 0 0 0 0 1 1 0 0
IV 0 0 0 0 0 0 0 0 0

Output/state values:
x 1 2 4 7 11 12 14 17 20
y 1 1 2 3 4 12 14 3 3
```

Simulation result

2.7.3   EdgeBi (EDGEBI)

2.7.3.1   Verbal description

The output indicates true at any change of the logical input value.

2.7.3.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 u: logic 2 R: logic 3 IV: logic	1 y: logic	1 x: logic	

2.7.3.3 Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	      x = IV;
	 }
	 y = (x != u);
	 x = u;

2.7.3.4 Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
u 1 0 0 1 1 0 1 0 1
R 0 0 0 0 0 0 1 1 0
IV 0 0 0 0 0 0 0 0 0

Output/state values:
x 1 0 0 1 1 0 1 0 1
y 1 1 0 1 0 1 1 0 1
```
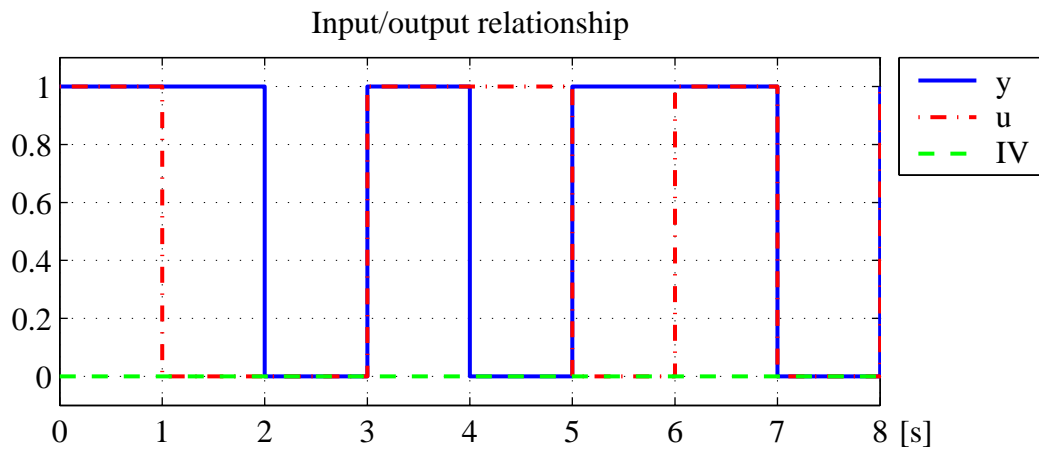
Simulation result

Control/reset input

Input/output relationship
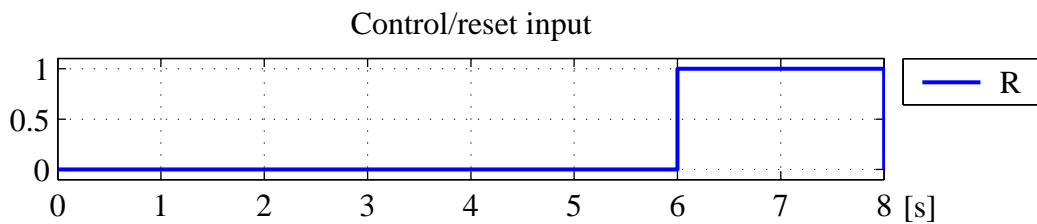
2.7.4 EdgeFalling (EDGEFALLING)

2.7.4.1 Verbal description

The output indicates true if the input value changes from true to false.

2.7.4.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 u: logic 2 R: logic 3 IV: logic	1 y: logic	1 x: logic	

2.7.4.3 Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	    x = IV;
	}
	y = (x && !u);
	x = u;

2.7.4.4 Test-cases

Testdata
Time vector:
```
t 0 1 2 3 4 5 6 7
```

Input values:
```
u 1 0 0 1 0 1 0 1
R 0 0 0 0 0 0 1 1 0
IV 0 0 0 0 0 0 0 0
```
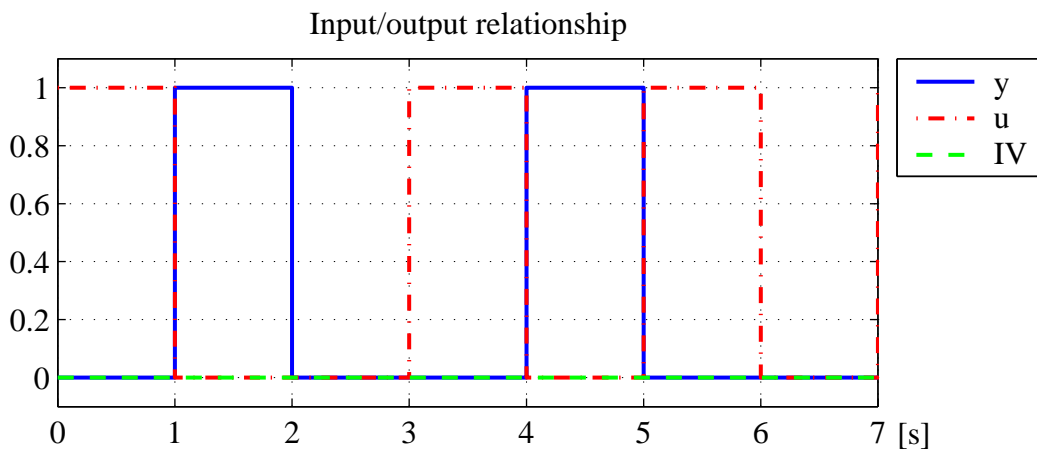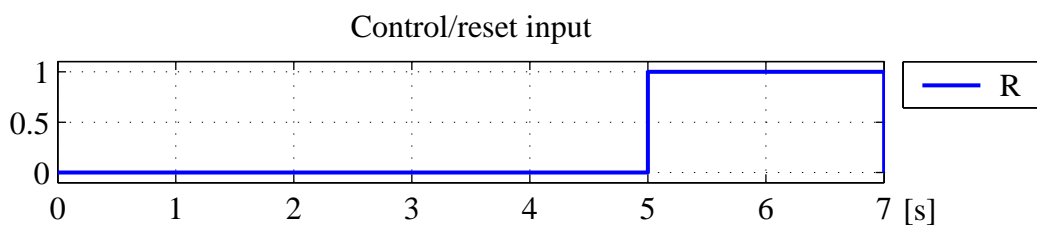
Output/state values:
```
x 1 0 0 1 0 1 0 1
y 0 1 0 0 1 0 0 0
```

Simulation result

2.7.5 EdgeRising (EDGERISING)

2.7.5.1 Verbal description

The output indicates true if the input value changes from false to true.

2.7.5.2   Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 u: logic 2 R: logic 3 IV: logic	1 y: logic	1 x: logic	

2.7.5.3   Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (R) {
	        x = IV;
	}
	y = (!x && u);
	x = u;

2.7.5.4   Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7

Input values:
u 1 1 0 1 1 0 1 0
R 0 0 0 0 0 1 1 0
IV 0 0 0 0 0 0 0 0

Output/state values:
x 1 1 0 1 1 0 1 0
y 1 0 0 1 0 0 1 0
```
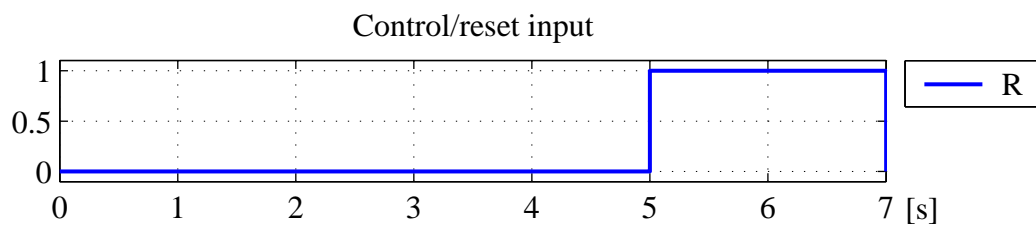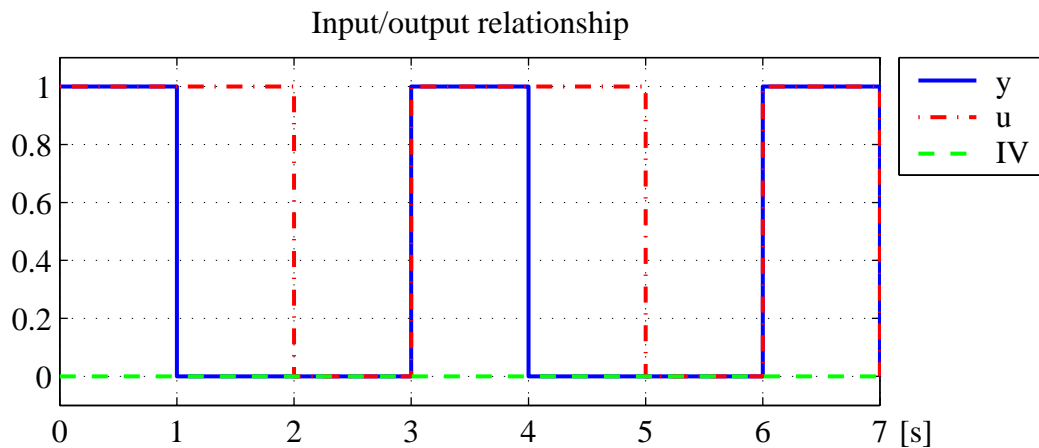
Simulation result

Control/reset input

Input/output relationship

2.7.6   RSFlipFlop (RSFLIPFLOP)

2.7.6.1   Verbal description

The second input $u2$ (reset) dominates the first input $u1$ (set). The first output returns the stored boolean input value, whilst the second output value is the logical negation of the first output.

$$y1(0) = false \qquad (2.10)$$

$$y1(n) = \begin{cases} false & : & u2(n) == true \\ true & : & u2(n) == false \&\& u1(n) == true \end{cases} \qquad (2.11)$$

$$y2(n) =!y1(n) \qquad (2.12)$$

2.7.6.2   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
R Q **F F** S Q̄	1 u1: logic 2 u2: logic	1 y1: logic 2 y2: logic	1 x: logic	

2.7.6.3   Pseudo-code

System-Init-Code	Run-Code

x = 0;	if (u2) {
	    x = 0;
	}
	else if (u1) {
	    x = 1;
	}
	y1 = x;
	y2 = !x;

2.7.6.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6 7

Input values:
u1 0 0 0 0 1 1 1 1
u2 1 1 0 0 0 0 1 1

Output/state values:
x 0 0 0 0 1 1 0 0
y1 0 0 0 0 1 1 0 0
y2 1 1 1 1 0 0 1 1
```
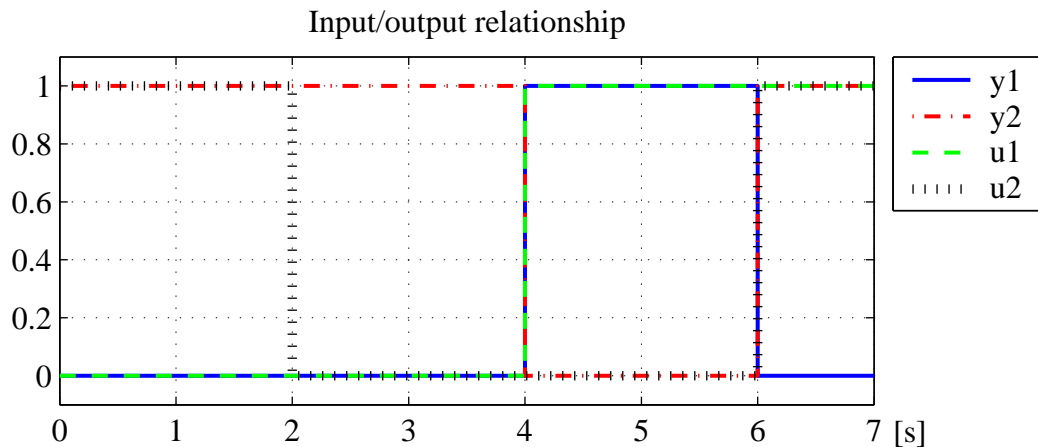
Simulation result

Input/output relationship

2.7.7   SampleAndHold_ResetEnabled (SAMPLEANDHOLD_RE)

2.7.7.1   Verbal description

Sample and hold or just a simple memory block with enable and reset ports.

2.7.7.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
S & H	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real	1 x: real	

2.7.7.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    x = IV;
	}
	else if (E) {
	    x = u;
	}
	y = x;

2.7.7.4 Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5 6 7

Input values:
u 1 2 4 7 11 12 14 17
E 0 0 1 1 1 1 0 1
R 0 0 0 0 0 1 1 0
IV 0 0 0 0 0 0 0 0

Output/state values:
x 0 0 4 7 11 0 0 17
y 0 0 4 7 11 0 0 17
```
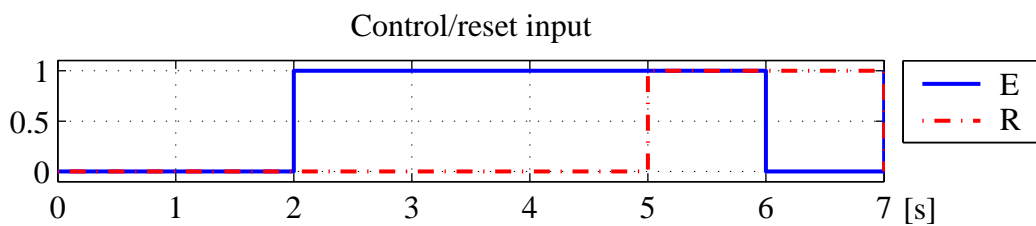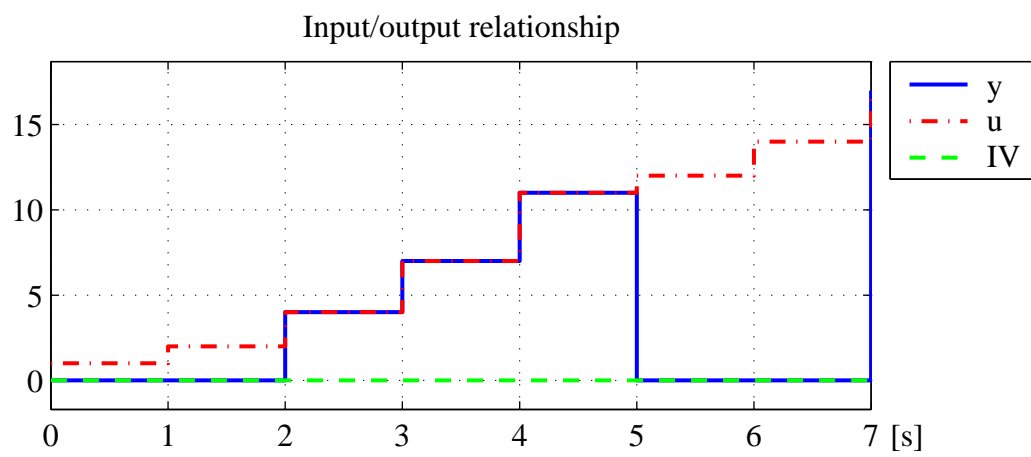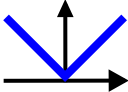
Simulation result

Control/reset input

Input/output relationship

2.8 Nonlinear blocks

2.8.1 AbsoluteValue (ABS)

2.8.1.1 Verbal description

$$y(n) = \begin{cases} u(n) & : & u(n) \geq 0 \\ -u(n) & : & u(n) < 0 \end{cases} \qquad (2.13)$$

2.8.1.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 u: real	1 y: real		

2.8.1.3 Pseudo-code

System-Init-Code	Run-Code

	if (u >= 0.0) {
	    y = u;
	}
	else {
	    y = -u;
	}

2.8.1.4 Test-cases

Testdata
Time vector:
t 0 1 2 3 4

Input values:
u -3 -1 0 1 3

Output/state values:
y 3 1 0 1 3

Simulation result
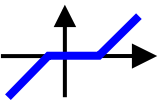
Input/output relationship
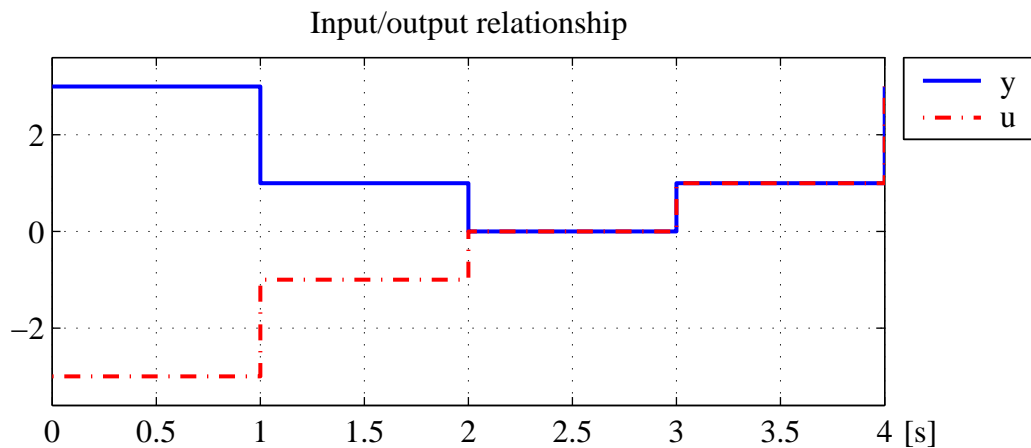
2.8.2 DeadBand (DEADBAND)

2.8.2.1 Verbal description

Returns zero, if the input value is between $UMIN$ and $UMAX$. Otherwise the output signal is the input signal reduced by the input limits:

$$y(n) = \begin{cases} 0.0 & : & u(n) < UMAX \,\&\& u(n) > UMIN \\ u(n) - UMAX & : & u(n) \geq UMAX \\ u(n) - UMIN & : & u(n) \leq UMIN \end{cases} \qquad (2.14)$$

2.8.2.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 UMAX: real 2 UMIN: real 3 u: real	1 y: real		

2.8.2.3 Pseudo-code

System-Init-Code	Run-Code

	if (u < UMAX && u > UMIN) {
	    y = 0.0;
	}
	else if (u >= UMAX) {
	    y = u - UMAX;
	}
	else {
	    y = u - UMIN;
	}

2.8.2.4 Test-cases

Testdata
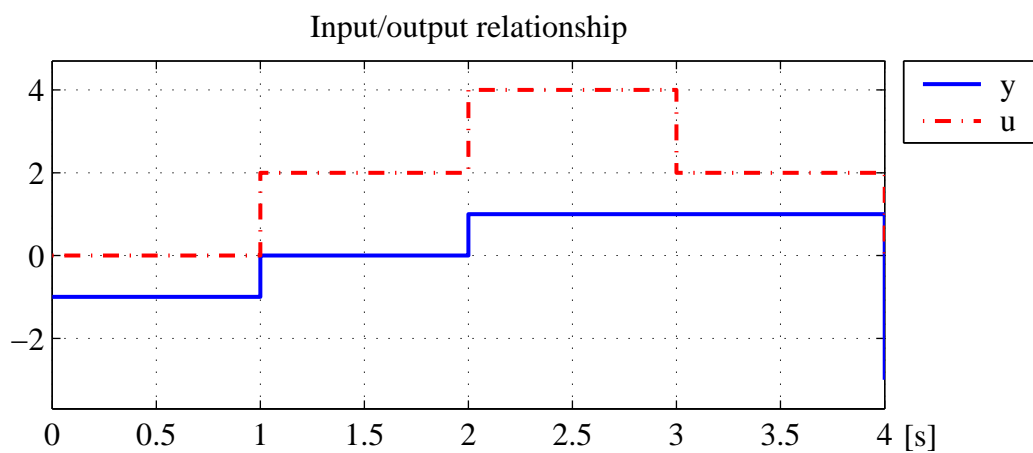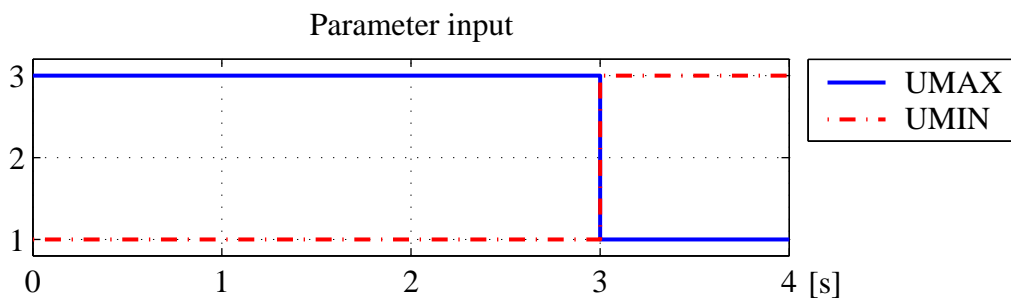```
Time vector:
t 0 1 2 3 4

Input values:
UMAX 3 3 3 1 1
UMIN 1 1 1 3 3
u 0 2 4 2 0

Output/state values:
y -1 0 1 1 -3
```

**Simulation result**



Parameter input



Input/output relationship

## 2.8.3   DifferenceLimiter (DIFFERENCELIMITER)

### 2.8.3.1   Verbal description

Increments the output value $y$ by the limited difference between consecutive input values $\Delta u = u(n) - u(n-1)$:

$$y(n) = \begin{cases} y(n-1) + LU & : & \Delta u > LU \\ u(n) & : & LU \geq \Delta u \geq LD \\ y(n-1) + LD & : & \Delta u < LU \end{cases} \qquad (2.15)$$

The input parameters can have signs and the condition $LU \geq LD$ is assumed, but not checked.

### 2.8.3.2   Icon and variables

| Icon | Variables | | | |
|---|---|---|---|---|
| | Inputs | Outputs | States | Temporary |
|  | 1  LU:  real<br>2  LD:  real<br>3  u:  real<br>4  E:  logic<br>5  R:  logic<br>6  IV:  real | 1  y:  real<br>2  B_max:  logic<br>3  B_min:  logic | 1  x:  real | |

### 2.8.3.3 Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x = 0.0; | if (R) {<br>    x = IV;<br>}<br>else if (E) {<br>    if ( (u - x) > LU ) {<br>        x = x + LU;<br>        B_min = 0;<br>        B_max = 1;<br>    }<br>    else if ( (u - x) < LD ) {<br>        x = x + LD;<br>        B_min = 1;<br>        B_max = 0;<br>    }<br>    else {<br>        x = u;<br>        B_min = 0;<br>        B_max = 0;<br>    }<br>}<br>y = x; |

### 2.8.3.4   Test-cases

**Testdata**
```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
LU 1 1 1 1 1 1 -3 1 1 1 1
LD 1 -1 -1 -1 -1 -3 1 -1 -1 -1 -1
u 0 2 4 4 2 2 0 2 4 2 0
E 1 1 1 1 1 1 1 1 0 1 1
R 0 0 0 0 0 0 0 1 1 0 0
IV 0 0 0 0 0 0 0 0 1 0 0

Output/state values:
x 1 2 3 4 3 2 -1 0 1 2 1
y 1 2 3 4 3 2 -1 0 1 2 1
B_min 1 0 0 0 1 0 0 0 0 0 1
B_max 0 0 1 0 0 0 1 1 1 0 0
```
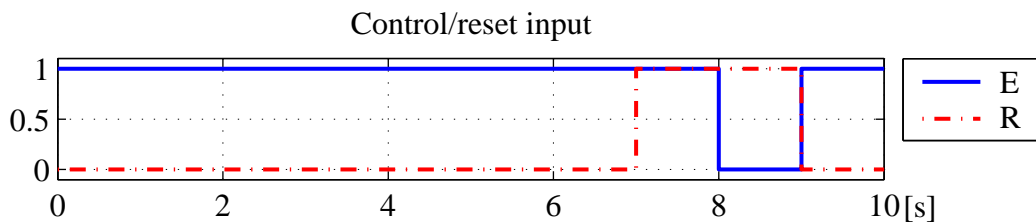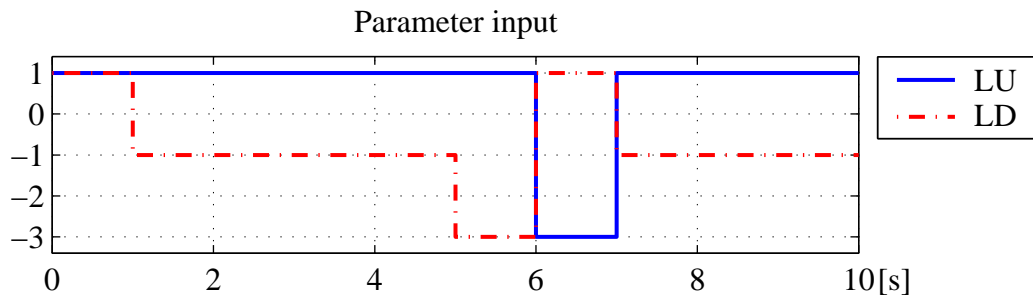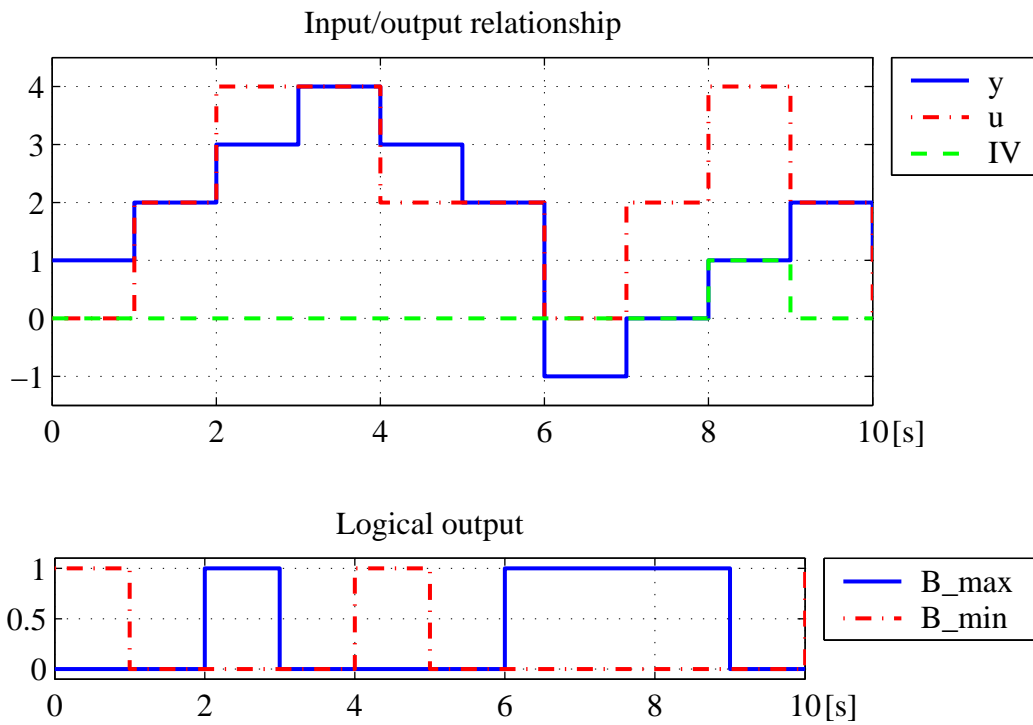
**Simulation result**



Parameter input



Control/reset input

Input/output relationship



Logical output



## 2.8.4 GradientLimiter (GRADIENTLIMITER)

### 2.8.4.1 Verbal description

Increments the output value $y$ by the limited difference between consecutive input values *over time* $\Delta u = \frac{u(n) - u(n-1)}{dT}$:

$$y(n) = \begin{cases} y(n-1) + LU & : & \Delta u > LU \\ u(n) & : & LU \geq \Delta u \geq LD \\ y(n-1) + LD & : & \Delta u < LU \end{cases} \qquad (2.16)$$

The input parameters can have signs and the condition $LU \geq LD$ is assumed, but not checked.

### 2.8.4.2 Icon and variables

| Icon | Variables | | | |
|------|-----------|---|---|---|
| | **Inputs** | **Outputs** | **States** | **Temporary** |
| | 1  LU:   real<br>2  LD:   real<br>3  u:    real<br>4  E:    logic<br>5  R:    logic<br>6  IV:   real | 1  y:       real<br>2  B_max:  logic<br>3  B_min:  logic | 1  x:  real | |

### 2.8.4.3   Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x = 0.0; | ```<br>if (R) {<br>    x = IV;<br>}<br>else if (E) {<br>    if ( (u - x)/dT > LU ) {<br>        x = x + LU;<br>        B_min = 0;<br>        B_max = 1;<br>    }<br>    else if ( (u - x)/dT < LD ) {<br>        x = x + LD;<br>        B_min = 1;<br>        B_max = 0;<br>    }<br>    else {<br>        x = u;<br>        B_min = 0;<br>        B_max = 0;<br>    }<br>}<br>y = x;<br>``` |

### 2.8.4.4 Test-cases

**Testdata**
Time vector:
```
t 0 1 2 3 4 5 6 7 8 9 10
```

Input values:
```
LU 1 1 1 1 1 1 -3 1 1 1 1
LD 1 -1 -1 -1 -1 -3 1 -1 -1 -1 -1
u 0 2 4 4 2 2 0 2 4 2 0
E 1 1 1 1 1 1 1 1 0 1 1
R 0 0 0 0 0 0 0 1 1 0 0
IV 0 0 0 0 0 0 0 0 1 0 0
```
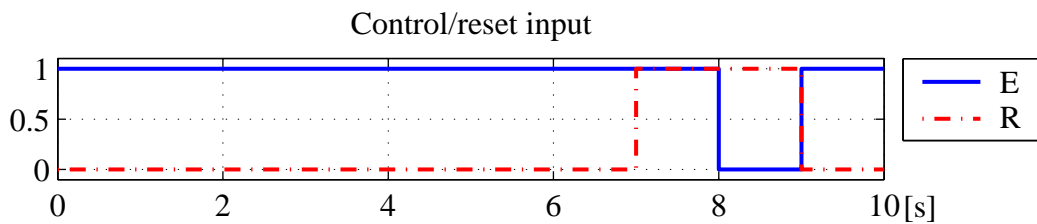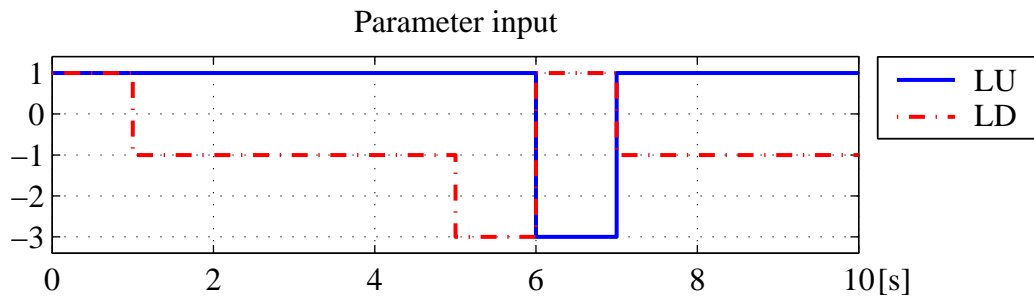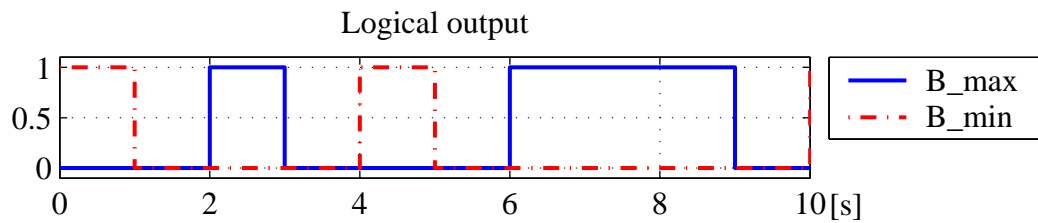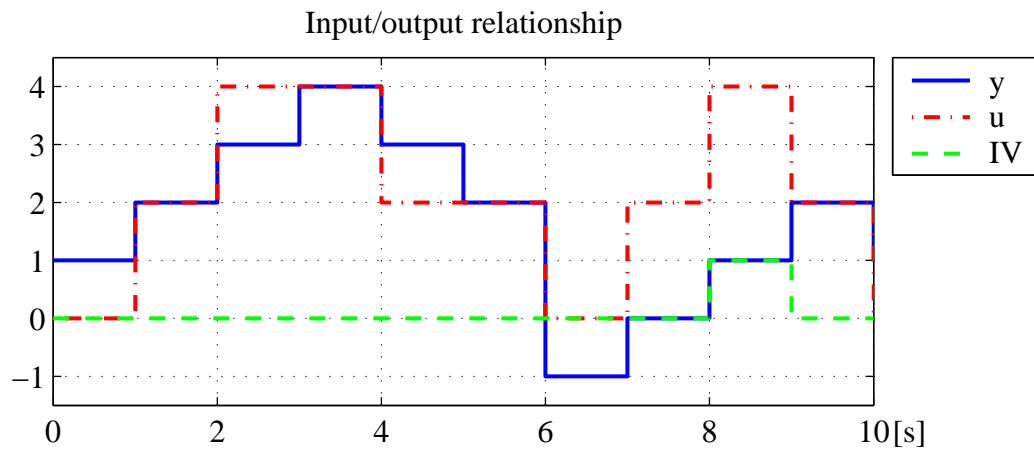
Output/state values:
```
x 1 2 3 4 3 2 -1 0 1 2 1
y 1 2 3 4 3 2 -1 0 1 2 1
B_min 1 0 0 0 1 0 0 0 0 0 1
B_max 0 0 1 0 0 0 1 1 1 0 0
```

**Simulation result**



Parameter input



Control/reset input

Input/output relationship



Logical output



## 2.8.5   Hysteresis (HYSTERESIS)

### 2.8.5.1   Verbal description

$$y(n) = \begin{cases} 1 & : & u > RSP \\ y(n-1) & : & RSP \geq u \geq LSP \\ 0 & : & u < LSP \end{cases} \tag{2.17}$$

### 2.8.5.2   Icon and variables

| Icon | Variables | | | |
|------|-----------|---|---|---|
| | Inputs | Outputs | States | Temporary |
| | 1  LSP:   real<br>2  RSP:   real<br>3  u:        real<br>4  R:        logic<br>5  IV:      logic | 1  y:   logic | 1  x:   logic | |

### 2.8.5.3   Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x = 0; | if (R) {<br>        x = IV;<br>}<br>if (u > RSP) {<br>        x = 1;<br>}<br>else if (u < LSP) {<br>        x = 0;<br>}<br>y = x; |

### 2.8.5.4   Test-cases

**Testdata**

```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
LSP 1 1 1 1 3 3 3 1 1 1 1
RSP 3 3 3 3 1 1 1 3 3 3 3
u 0 2 4 2 0 2 4 2 0 4 4
R 0 0 0 0 0 0 0 1 0 1 0
IV 0 0 0 0 0 0 0 0 0 0 0

Output/state values:
x 0 0 1 1 0 1 1 0 0 1 1
y 0 0 1 1 0 1 1 0 0 1 1
```
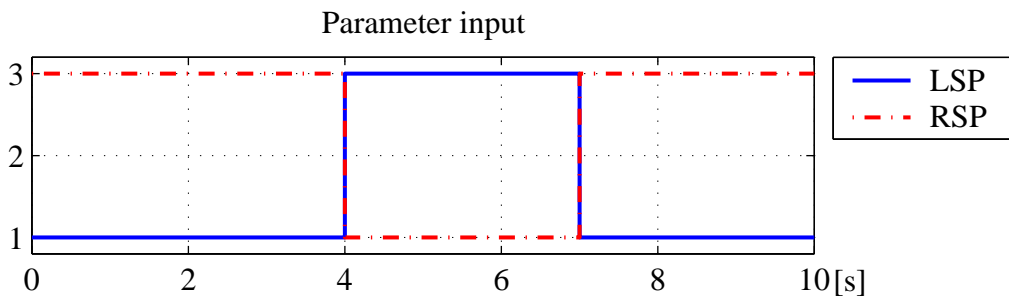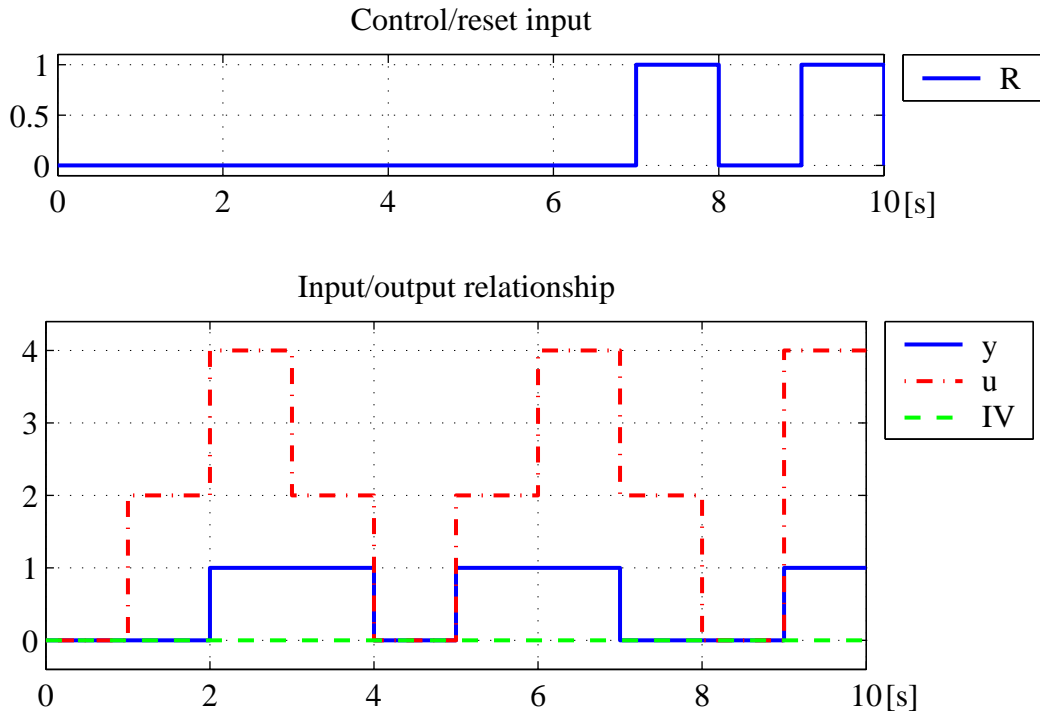
**Simulation result**



Parameter input

Control/reset input



Input/output relationship



## 2.8.6  Limiter (LIMIT)

### 2.8.6.1  Verbal description

The output value is the limited input value:

$$y(n) = \begin{cases} MX & : & u(n) > MX \\ MN & : & u(n) < MN \&\& u(n) \leq MX \\ u(n) & : & MN \leq u(n) \leq MX \end{cases} \quad (2.18)$$

The increasing value domain is splitted in different intervals depending on the limitation values $MN, MX$ (see figure 2.1).
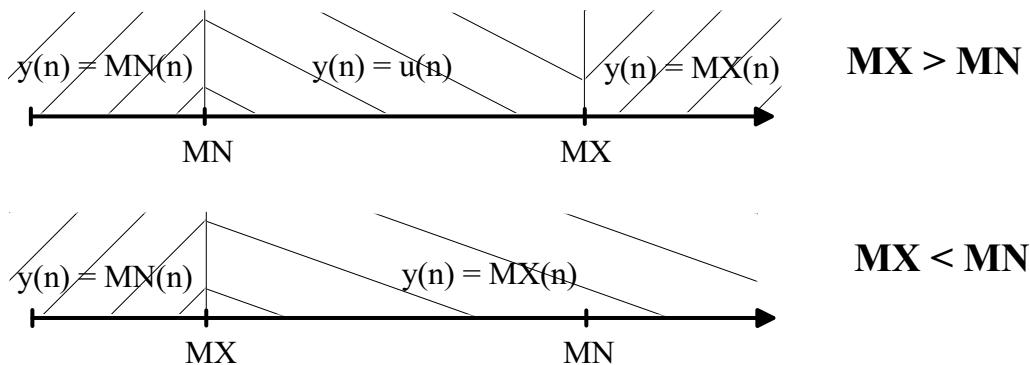


Figure 2.1: Intervals for different limitation values.

The boolean output flags $B\_MAX$ and $B\_MIN$ represent an active limitation in both directions.

### 2.8.6.2 Icon and variables

| Icon | Variables | | | | |
|------|-----------|---|---|---|---|
| | **Inputs** | | **Outputs** | | **States** | **Temporary** |

| Icon | Inputs | | | Outputs | | | States | Temporary |
|------|---|---|---|---|---|---|---|---|
| | 1 | MX: | real | 1 | y: | real | | |
| | 2 | MN: | real | 2 | B_max: | logic | | |
| | 3 | u: | real | 3 | B_min: | logic | | |

### 2.8.6.3 Pseudo-code

| System-Init-Code | Run-Code |
|------------------|----------|
| | ```
if (u > MX) {
    y = MX;
    B_max = 1;
    B_min = 0;
}
else if (u < MN) {
    y = MN;
    B_max = 0;
    B_min = 1;
}
else {
    y = u;
    B_max = 0;
    B_min = 0;
}
``` |

2.8.6.4 Test-cases

Testdata

```
Time vector:
t 0 1 2 3 4 5

Input values:
MX 2 2 2 0 0 0
MN 0 0 0 2 2 -2
u -1 1 3 3 1 -1

Output/state values:
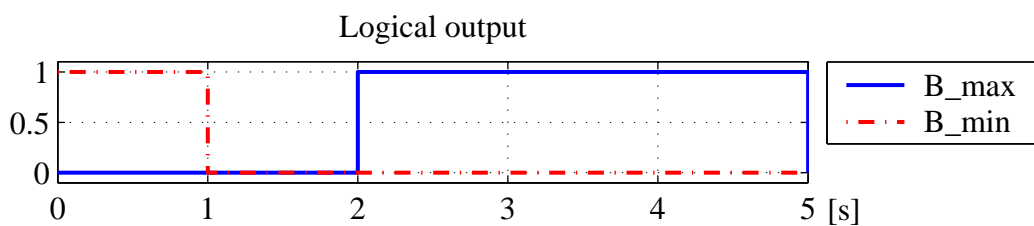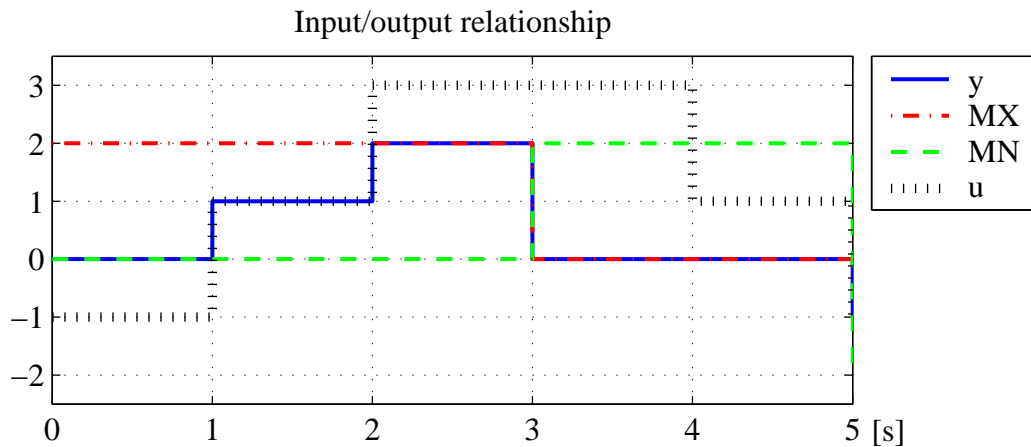y 0 1 2 0 0 -1
B_min 1 0 0 0 0 0
B_max 0 0 1 1 1 0
```

Simulation result

Input/output relationship

Logical output

2.8.7   Maximum (MAX)

2.8.7.1   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
MAX	1 ui: real	1 y: real		

2.8.7.2 Pseudo-code

System-Init-Code	Run-Code
	y = max(u1,u2,...,un);

2.8.8 MaxLogResetEnabled (MAXLOG_RE)

2.8.8.1 Verbal description

The output value is the maximum value of all occurred input values u with respect to the initial value IV.

2.8.8.2 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
MAX LOG	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real 2 B_max: logic	1 x: real	

2.8.8.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	B_max = 0;
	if (R) {
	    x = IV;
	}
	else if (E) {
	    if (u > x) {
	        x = u;
	        B_max = 1;
	    }
	}
	y = x;

2.8.8.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6

Input values:
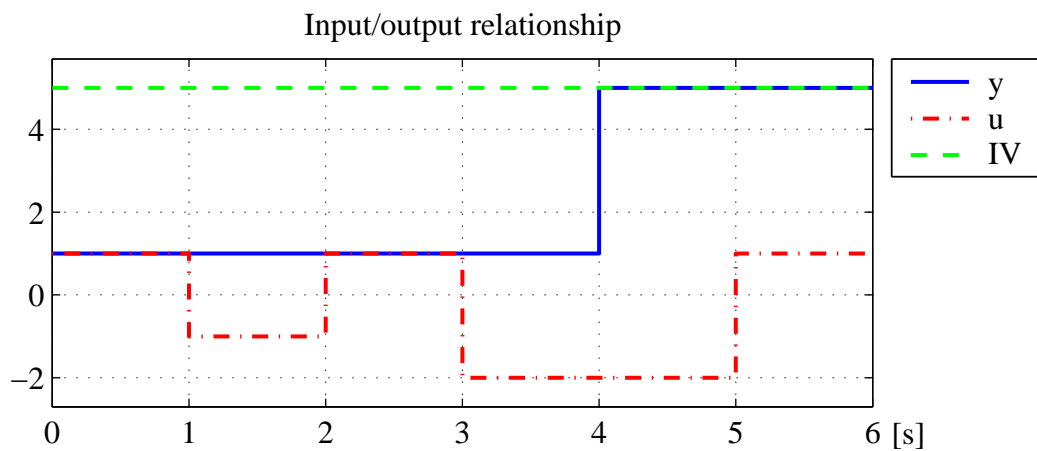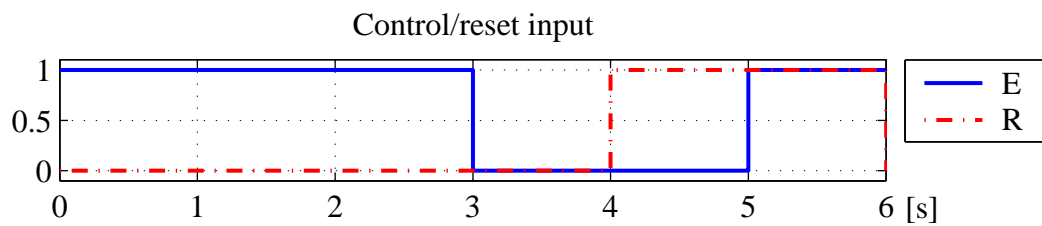u 1 -1 1 -2 -2 1 1
E 1 1 1 0 0 1 1
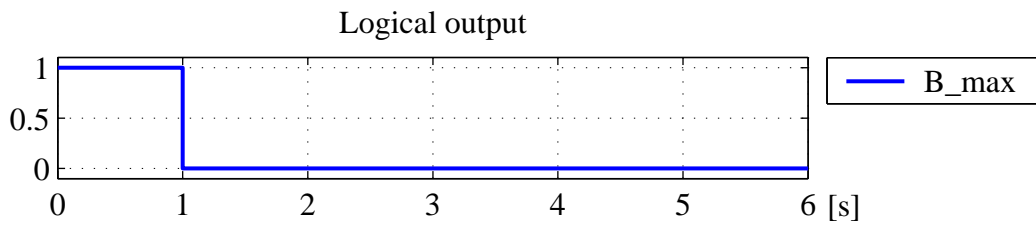R 0 0 0 0 1 1 0
IV 5 5 5 5 5 5 5

Output/state values:
x 1 1 1 1 5 5 5
y 1 1 1 1 5 5 5
R_max 1 0 0 0 0 0 0
```

Simulation result

Control/reset input

Input/output relationship

Logical output

2.8.9 MaxLogResetEnabled (MAXLOG_RE)

2.8.9.1 Verbal description

The output value is the maximum value of all occurred input values u with respect to the initial value IV.

2.8.9.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
MAX **LOG**	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real 2 B_max: logic	1 x: real	

2.8.9.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	B_max = 0;
	 if (R) {
	      x = IV;
	 }
	 else if (E) {
	      if (u > x) {
	          x = u;
	          B_max = 1;
	      }
	 }
	 y = x;

2.8.9.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6

Input values:
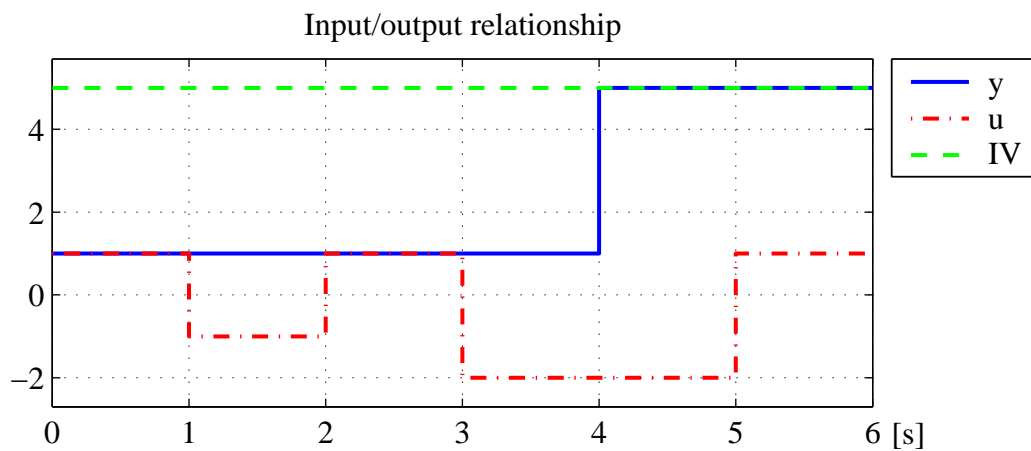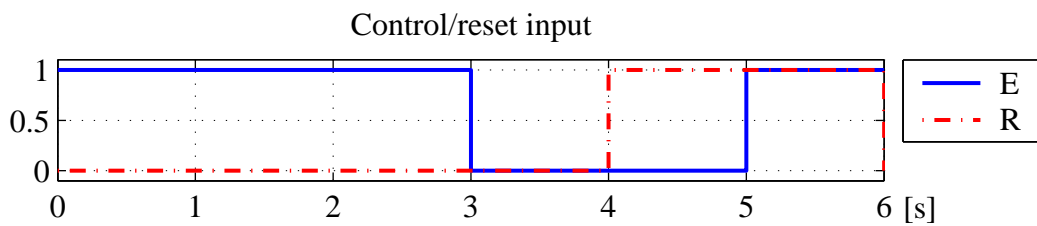u 1 -1 1 -2 -2 1 1
E 1 1 1 0 0 1 1
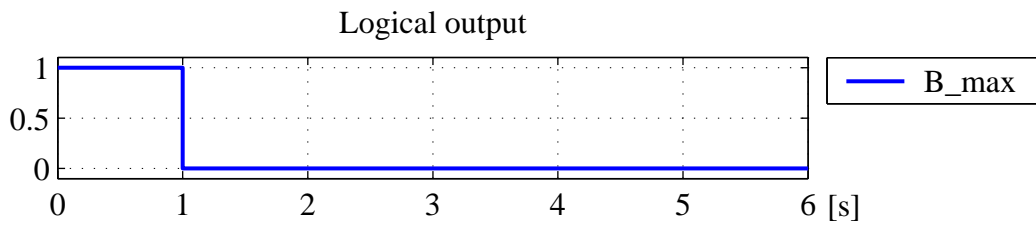R 0 0 0 0 1 1 0
IV 5 5 5 5 5 5 5

Output/state values:
x 1 1 1 1 5 5 5
y 1 1 1 1 5 5 5
R_max 1 0 0 0 0 0 0
```

Simulation result

Control/reset input

Input/output relationship

Logical output

2.8.10 MeanValue (MEANVALUE)

2.8.10.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
$\dfrac{\sum u_k}{K}$	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real	1 x: real 2 mean: real	1 i: uint

2.8.10.2 Pseudo-code

System-Init-Code	Run-Code

for(i = 0; i < k; i++){
    x[i] = 0.0;
}
mean = 0;
	if (R) {
	    for(i = 0; i < k; i++) {
	        x[i] = IV;
	    }
	    mean = IV;
	}
	else if (E) {
	    for(i = k - 1; i > 0; i–) {
	        x[i] = x[i-1];
	    }
	    mean = 0;
	    x[0]=u;
	    for(i = 0; i < k; i++) {
	        mean = mean + x[i];
	    }
	    mean = mean/k;
	}
	y = mean;

2.8.10.3   Test-cases

Testdata
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
u 1 3 5 7 9 7 5 3 1
E 1 1 1 1 1 0 0 1 1
R 0 0 0 0 0 0 1 0 1
IV 1 1 1 1 1 1 1 1 1

Output/state values:
mean 0.25 1 2.25 4 6 6 1 1.5 1
y 0.25 1 2.25 4 6 6 1 1.5 1

Simulation result

Control/reset input

Input/output relationship

2.8.11 Minimum (MIN)

2.8.11.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
MIN	1 ui: real	1 y: real		

2.8.11.2 Pseudo-code

System-Init-Code	Run-Code
	y = min(u1,u2,...,un);

2.8.12 MinLogResetEnabled (MINLOG_RE)

2.8.12.1 Verbal description

The output value is the minimum value of all occured input values u with respect to the initial value IV.

2.8.12.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
MIN **LOG**	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real 2 B_min: logic	1 x: real	

2.8.12.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	B_min = 0;
	 if (R) {
	     x = IV;
	 }
	 else if (E) {
	     if (u < x) {
	         x = u;
	         B_min = 1;
	     }
	 }
	 y = x;

2.8.12.4   Test-cases

Testdata
```
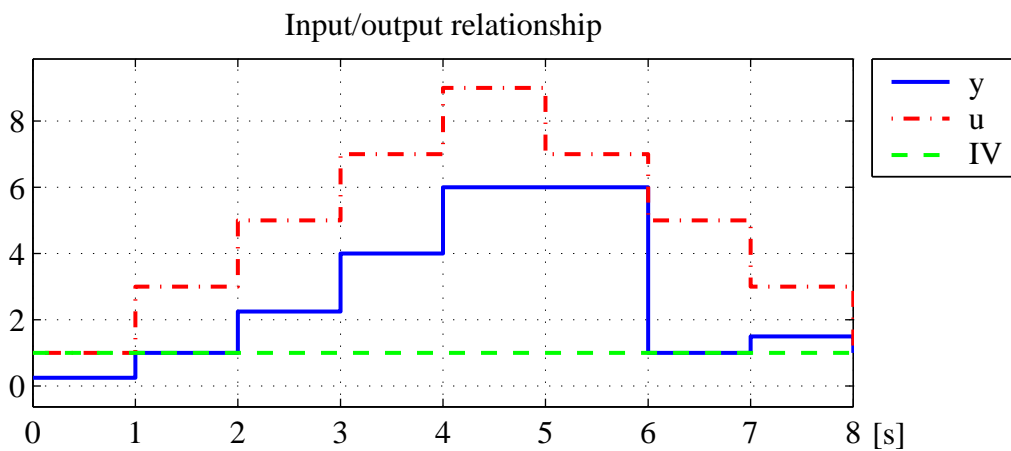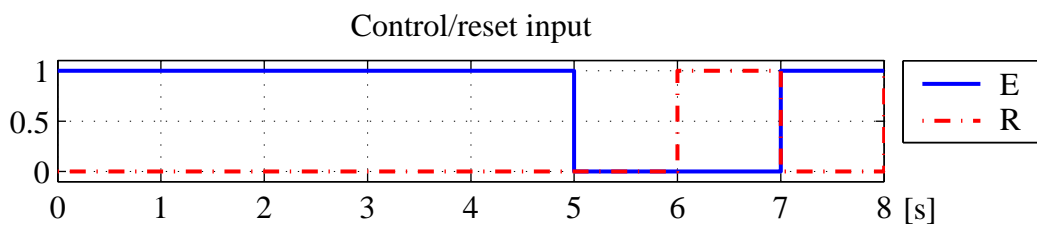Time vector:
t 0 1 2 3 4 5 6

Input values:
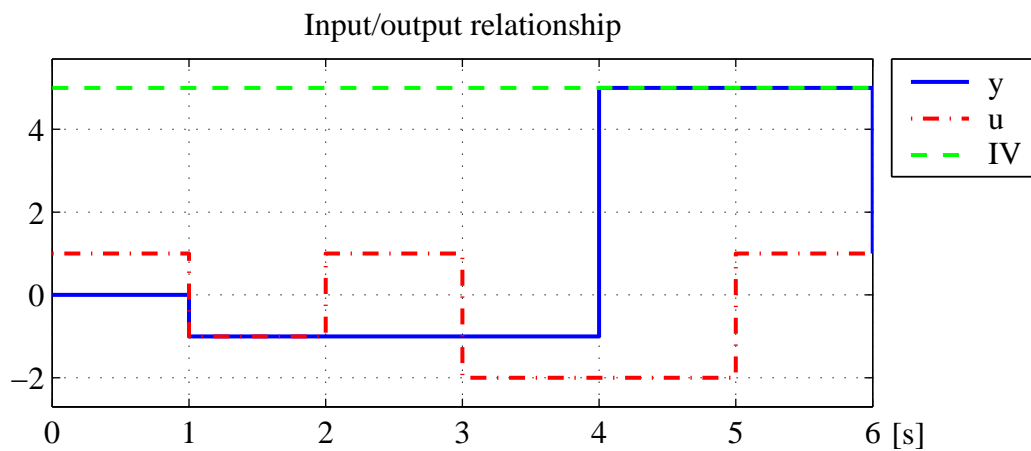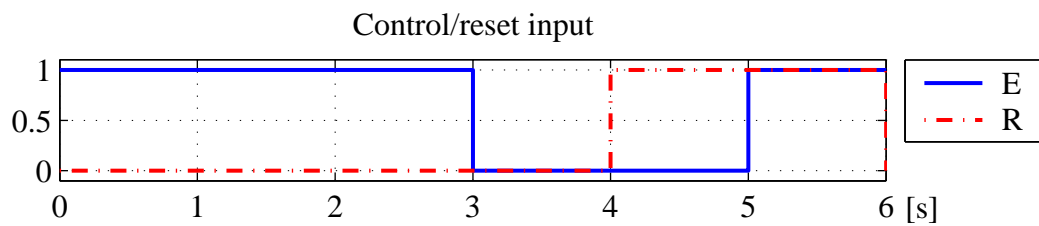u 1 -1 1 -2 -2 1 1
E 1 1 1 0 0 1 1
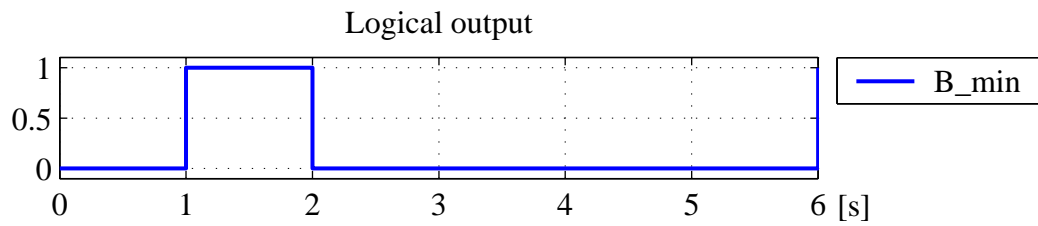R 0 0 0 0 1 1 0
IV 5 5 5 5 5 5 5

Output/state values:
x 0 -1 -1 -1 5 5 1
y 0 -1 -1 -1 5 5 1
R_min 0 1 0 0 0 0 1
```

Simulation result

Control/reset input

Input/output relationship

Logical output

2.8.13 MinLogResetEnabled (MINLOG_RE)

2.8.13.1 Verbal description

The output value is the minimum value of all occured input values u with respect to the initial value IV.

2.8.13.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
MIN **LOG**	1 u: real 2 E: logic 3 R: logic 4 IV: real	1 y: real 2 B_min: logic	1 x: real	

2.8.13.3 Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	B_min = 0;
	if (R) {
	    x = IV;
	}
	else if (E) {
	    if (u < x) {
	        x = u;
	        B_min = 1;
	    }
	}
	y = x;

2.8.13.4   Test-cases

Testdata
Time vector:
t 0 1 2 3 4 5 6

Input values:
u 1 -1 1 -2 -2 1 1
E 1 1 1 0 0 1 1
R 0 0 0 0 1 1 0
IV 5 5 5 5 5 5 5

Output/state values:
x 0 -1 -1 -1 5 5 1
y 0 -1 -1 -1 5 5 1
R_min 0 1 0 0 0 0 1

Simulation result

Control/reset input

Input/output relationship

Logical output

2.8.14 Signum (SIG)

2.8.14.1 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 u: real	1 y: real		

2.8.14.2 Pseudo-code

System-Init-Code	Run-Code

	if (u > 0.0) {
	    y = 1.0;
	}
	else if (u < 0.0) {
	    y = -1.0;
	}
	else {
	    y = 0.0;
	}

2.8.14.3 Test-cases

Testdata
Time vector:
t 0 1 2

Input values:
u -2 0 2

Output/state values:
y -1 0 1

Simulation result

Input/output relationship

2.9 Integrators

2.9.1 AccumulatorResetEnabledLimited (ACCUMULATOR_REL)

2.9.1.1 Verbal description

The output y is the limited sum, stored in x, of all input values u. IV is the initial value of the sum.

2.9.1.2 Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
$\sum$	1 MX: real 2 MN: real 3 u: real 4 E: logic 5 R: logic 6 IV: real	1 y: real 2 B_max: logic 3 B_min: logic	1 x: real	

2.9.1.3 Pseudo-code

System-Init-Code	Run-Code
| x = 0.0; | ```
if (R) {
    x = IV;
}
else if (E) {
    x = x + u;
}
if (x > MX) {
    x = MX;
    B_max = 1;
    B_min = 0;
}
else if (x < MN) {
    x = MN;
    B_max = 0;
    B_min = 1;
}
else {
    B_max = 0;
    B_min = 0;
}
y = x;
``` |
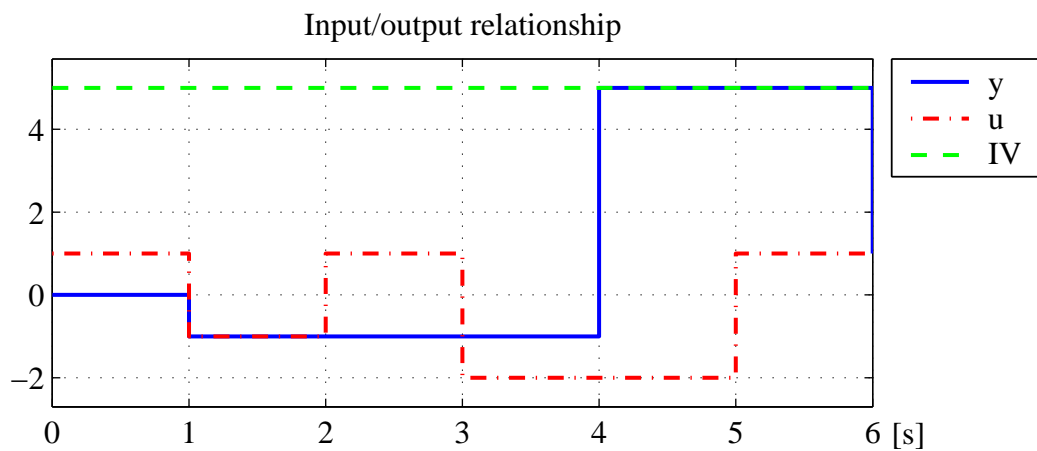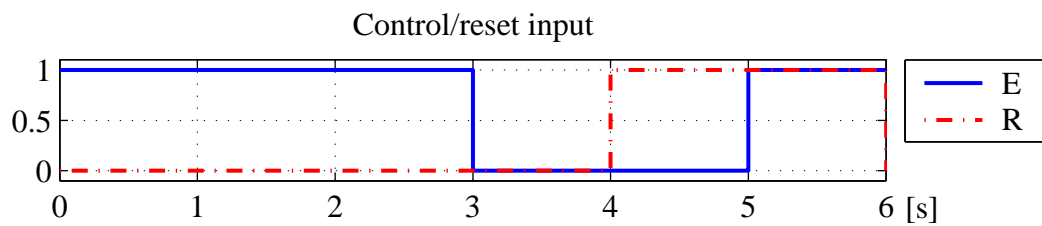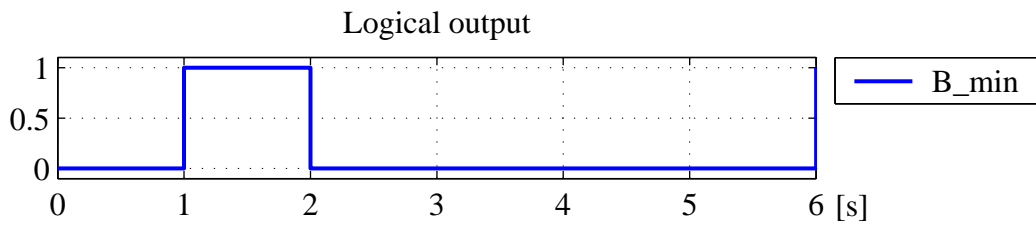
2.9.1.4   Test-cases
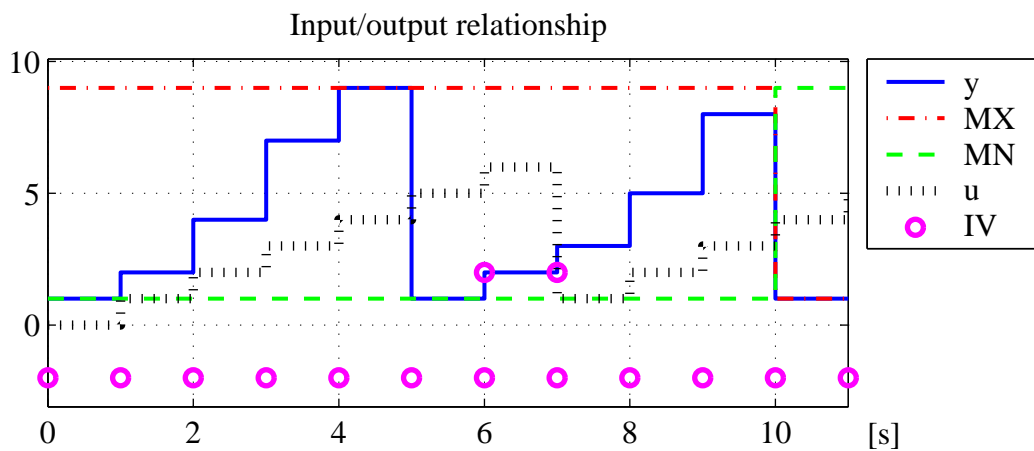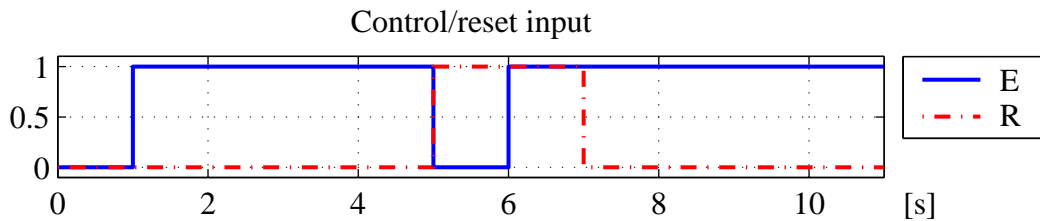
Testdata
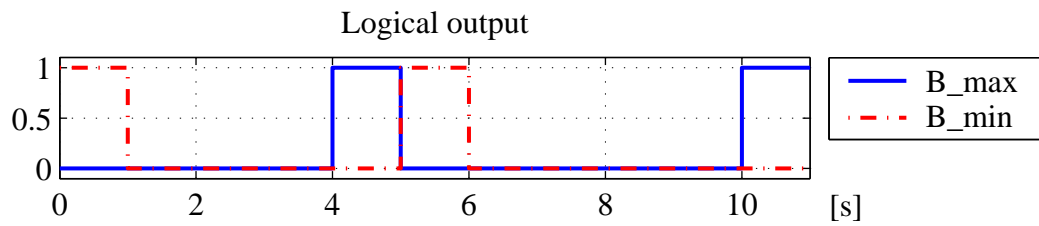```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10 11
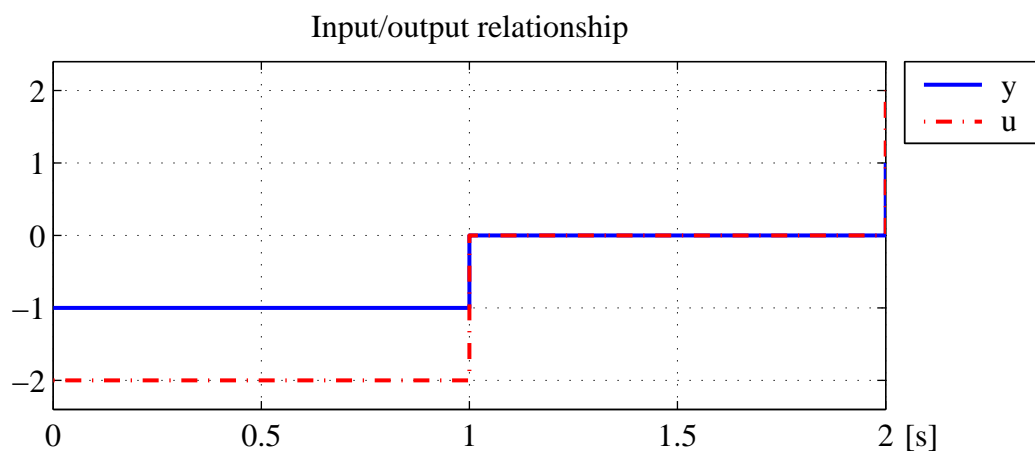
Input values:
MX 9 9 9 9 9 9 9 9 9 9 1 1
MN 1 1 1 1 1 1 1 1 1 1 9 9
u 0 1 2 3 4 5 6 1 2 3 4 5
E 0 1 1 1 1 0 1 1 1 1 1 1
R 0 0 0 0 0 1 1 0 0 0 0 0
IV -2 -2 -2 -2 -2 -2 2 -2 -2 -2 -2 -2

Output/state values:
x 1 2 4 7 9 1 2 3 5 8 1 1
y 1 2 4 7 9 1 2 3 5 8 1 1
B_min 1 0 0 0 0 1 0 0 0 0 0 0
B_max 0 0 0 0 1 0 0 0 0 0 1 1
```

**Simulation result**



Control/reset input



Input/output relationship

Logical output



## 2.9.2 IntegratorKResetEnabledLimited (INTEGRATORK_REL)

### 2.9.2.1 Verbal description

A time discrete integrator with gain K. The integrated value and therefore the output y are limited by the inputs MX and MN ($y \in [MN, MX]$).

### 2.9.2.2 Icon and variables

| Icon | Variables | | | |
|------|-----------|--|--|--|
| | **Inputs** | **Outputs** | **States** | **Temporary** |
| | 1  MX:  real<br>2  MN:  real<br>3  K:   real<br>4  u:   real<br>5  E:   logic<br>6  R:   logic<br>7  IV:  real | 1  y:     real<br>2  B_max: logic<br>3  B_min: logic | 1  x:  real | |

### 2.9.2.3    Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x = 0.0; | ```
if (R) {
    x = IV;
}
else if (E) {
    x = x + K*u*dT;
}
if (x > MX) {
    x = MX;
    B_max = 1;
    B_min = 0;
}
else if (x < MN) {
    x = MN;
    B_max = 0;
    B_min = 1;
}
else {
    B_max = 0;
    B_min = 0;
}
y = x;
``` |

2.9.2.4 Test-cases

Testdata
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10

Input values:
MX 5 5 5 5 5 1 1 5 5 5 5
MN 1 1 1 1 1 5 -1 1 1 1 1
K 2 2 2 -3 -3 2 2 2 2 2 2
u 1 1 1 1 1 1 1 1 1 1 1
E 1 1 1 1 1 1 0 0 1 1
R 0 0 0 0 0 0 0 0 1 0 1
IV 0 0 0 0 0 0 0 0 10 0 0

Output/state values:
x 2 4 5 2 1 1 1 1 5 5 1
y 2 4 5 2 1 1 1 1 5 5 1
B_min 0 0 0 0 1 0 0 0 0 0 1
B_max 0 0 1 0 0 1 1 0 1 1 0

Simulation result

Parameter input

Control/reset input

Input/output relationship

Logical output

2.9.3 IntegratorTResetEnabledLimited (INTEGRATORT_REL)

2.9.3.1 Verbal description

A time discrete integrator with gain $1/T$. The integrated value and therefore the output y are limited by the inputs MX and MN ($y \in [MN, MX]$).

2.9.3.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 MX: real 2 MN: real 3 T: real 4 u: real 5 E: logic 6 R: logic 7 IV: real	1 y: real 2 B_max: logic 3 B_min: logic	1 x: real	

2.9.3.3   Pseudo-code

System-Init-Code	Run-Code
| x = 0.0; | ```
if (R) {
    x = IV;
}
else if (E) {
    x = x + u*dT/T;
}
if (x > MX) {
    x = MX;
    B_max = 1;
    B_min = 0;
}
else if (x < MN) {
    x = MN;
    B_max = 0;
    B_min = 1;
}
else {
    B_max = 0;
    B_min = 0;
}
y = x;
``` |
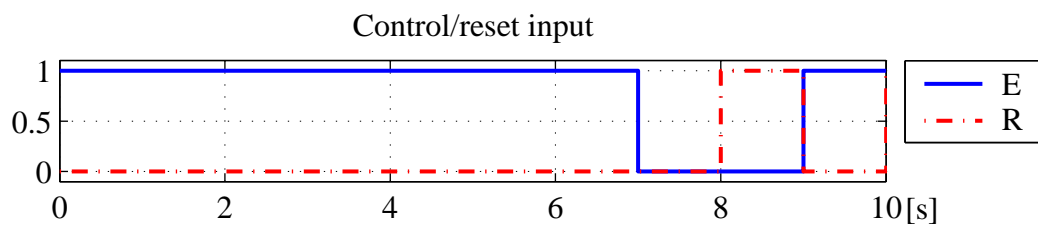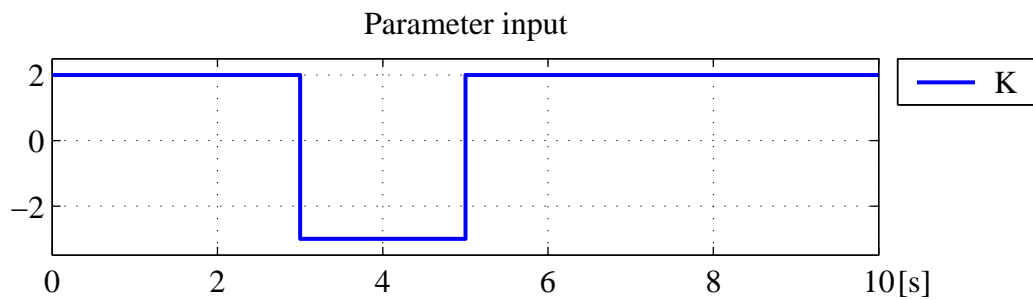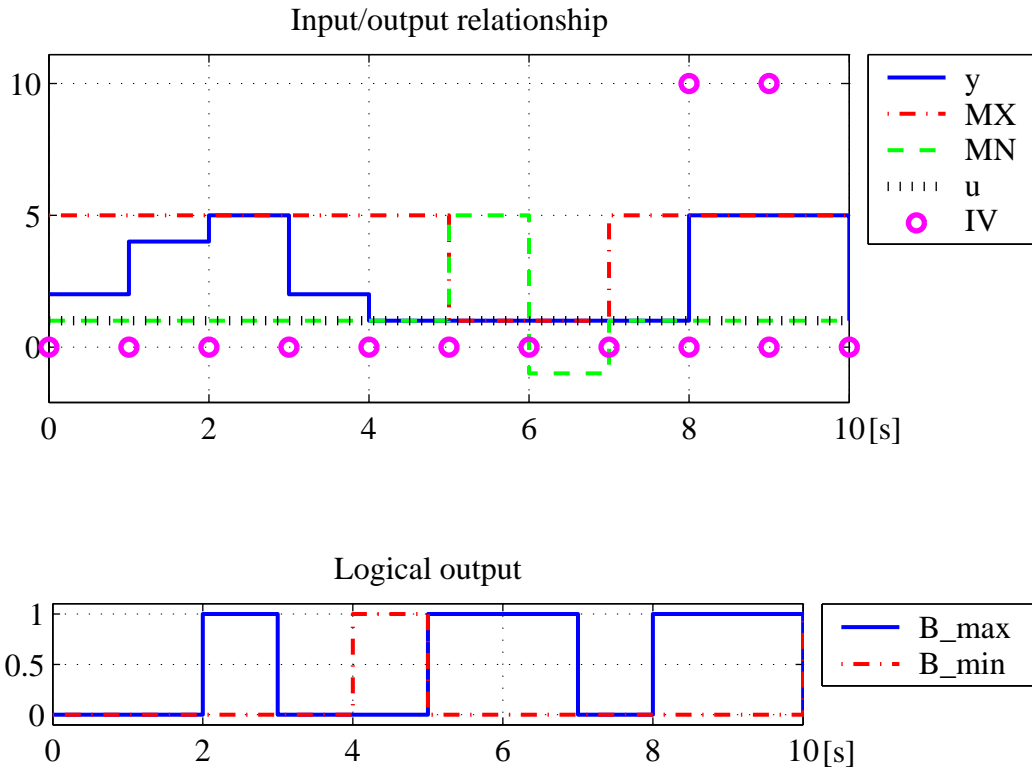
2.9.3.4   Test-cases

Testdata
Time vector:
```
t 0 1 2 3 4 5 6 7 8 9 10
```

Input values:
```
MX 5 5 5 5 5 1 1 5 5 5 5
MN 1 1 1 1 1 5 -1 1 1 1 1
T 0.5 0.5 0.5 -0.25 -0.25 0.5 0.5 0.5 0.5 0.5 0.5
u 1 1 1 1 1 1 1 1 1 1 1
E 1 1 1 1 1 1 1 0 0 1 1
R 0 0 0 0 0 0 0 0 1 0 1
IV 0 0 0 0 0 0 0 0 10 0 0
```
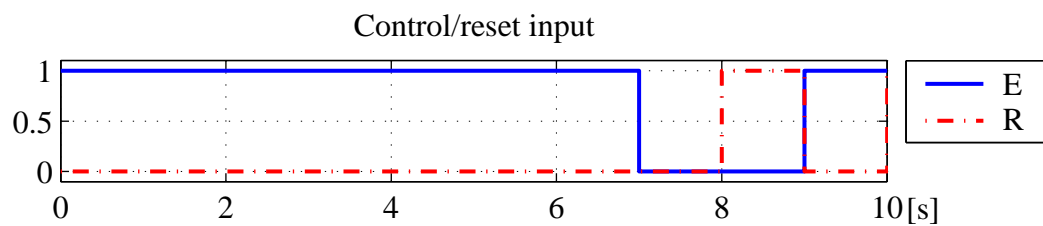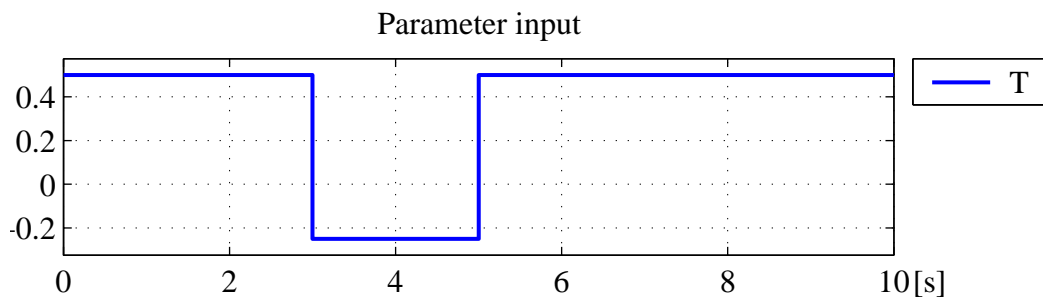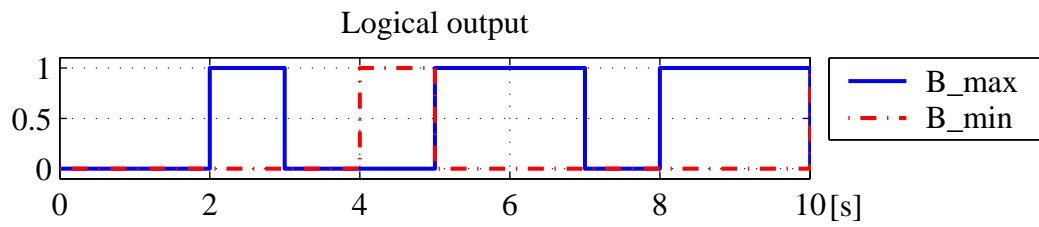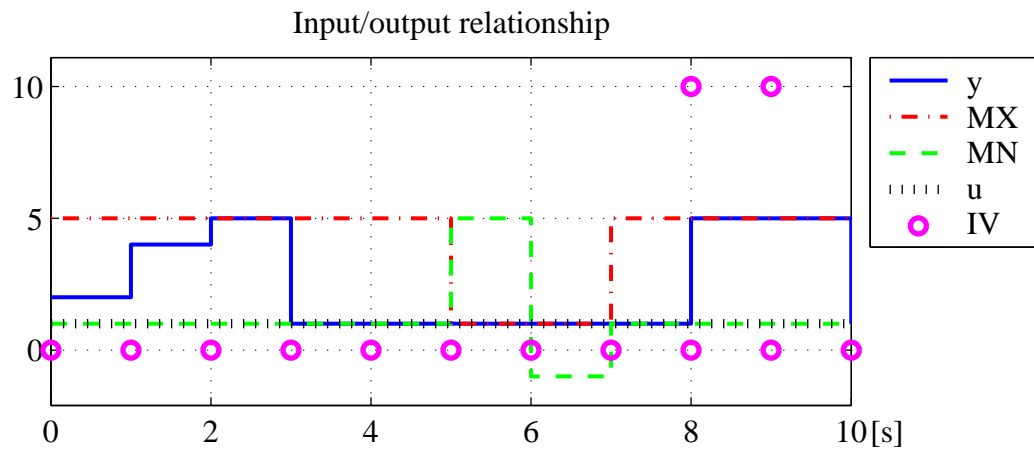
Output/state values:
```
x 2 4 5 1 1 1 1 1 5 5 1
y 2 4 5 1 1 1 1 1 5 5 1
B_min 0 0 0 0 1 0 0 0 0 0 1
B_max 0 0 1 0 0 1 1 0 1 1 0
```

Simulation result

Parameter input

Control/reset input

Input/output relationship

Logical output

2.10   Low- and highpass

In this section basic control elements are presented. The verbal description introduces the time continuous functions of the blocks. To convert time continuous functions to discrete functions the continuous description of the blocks is transferred to a discrete time functions using a 1st order approximation for the derivative $\dot{x}(t)$ in the differential equations (forward difference algorithm). The integration

$$y(t) = \frac{dx}{dt} \tag{2.19}$$

is sampled at period dT with $t = n \cdot dT$. The result is the difference

$$y(n-1) = \frac{x(n) - x(n-1)}{dT} \quad ; n = 0(1)\inf \tag{2.20}$$

$$\frac{y(n-1) - y(n-2)}{dT} = \frac{x(n) - 2 \cdot x(n-1) + x(n-2)}{dT^2} \quad ; n = 0(1)\inf . \tag{2.21}$$

Using this equation recursively allows the definition of higher order derivatives.
For the complex conversion of transfer functions from continuous (Laplace, s-domain) to discrete time domain (z-domain) results therefore

$$s = \frac{z-1}{dT}. \tag{2.22}$$

The relationship between the analogue ω_A and digital ω_D frequencies with $\omega_i = 2 \cdot \pi \cdot f_i, i \in (A, D)$ is

$$\omega_A = \frac{\sin(\omega_D \cdot dT)}{dT}. \tag{2.23}$$

2.10.1   DigitalLowpassResetEnabled (DIGITALLOWPASS_RE)

2.10.1.1   Verbal description

A discrete time first order lowpass with linear approximation according (see 2.10 on page 100). Assuming a transfer function in the frequency domain G(s) with

$$G(s) = \frac{K}{1 + T_1 \cdot s}. \tag{2.24}$$
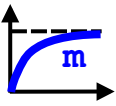
The relationship between m and T_1 or K is

$$T_1 = \frac{dT}{m} or K = m. \tag{2.25}$$

The input m indicates the percentage of the current approximation of the output value to the aimed input value. The relationship between the cut-off-frequency (f_c) and m is:

$$m = 2 \cdot \pi \cdot f_c \cdot dT \tag{2.26}$$

with sample time dT.

2.10.1.2   Icon and variables

Icon	Variables			
	Inputs	Outputs	States	Temporary
	1 m: real 2 u: real 3 E: logic 4 R: logic 5 IV: real	1 y: real	1 x: real	

2.10.1.3   Pseudo-code

System-Init-Code	Run-Code

x = 0.0;	if (R) {
	    x = IV;
	}
	else if (E) {
	    x = x + m*(u - x);
	}
	y = x;

2.10.1.4   Test-cases

Testdata
```
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
m 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
u 1 1 1 1 1 1 1 1 1
E 1 1 1 0 1 1 1 0 0
R 0 0 0 1 1 0 0 0 0
IV 0 0 0 0 0 0 0 0 0

Output/state values:
x 0.5 0.75 0.875 0 0 0.5 0.75 0.75 0.75
y 0.5 0.75 0.875 0 0 0.5 0.75 0.75 0.75
```
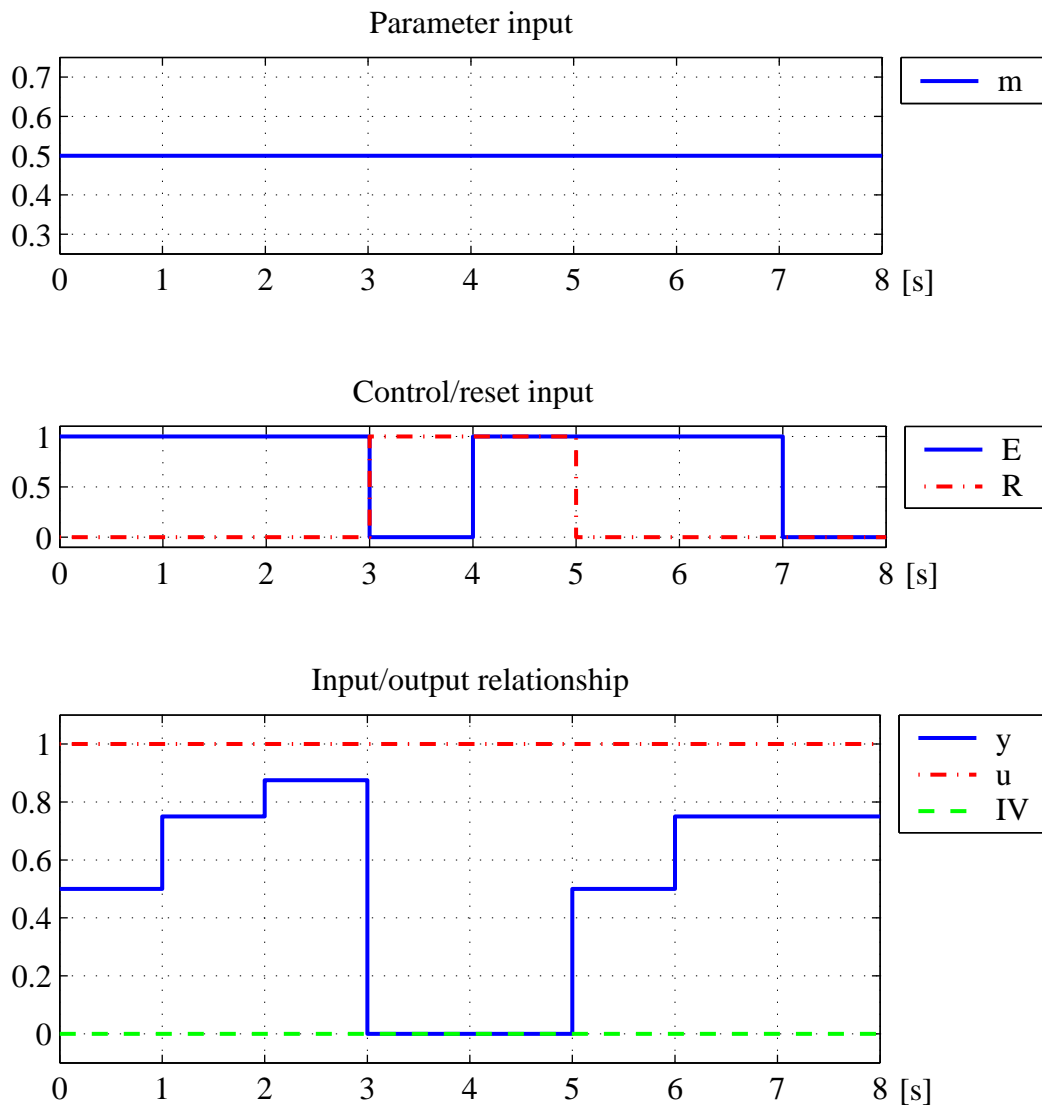
Simulation result

Parameter input

Control/reset input

Input/output relationship

2.10.2 HighpassTResetEnabled (HIGHPASST_RE)
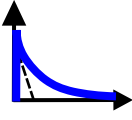
2.10.2.1   Verbal description

A discrete time first order highpass filter with the continuous time Laplace notation

$$G(s) = \frac{TD \cdot s}{(1 + T \cdot s)}. \tag{2.27}$$

The relationship between the analogue cut-off-frequency (f_c) and T is

$$f_c = \frac{1}{2 \cdot \pi \cdot T}. \tag{2.28}$$

2.10.2.2 Icon and variables

Icon	Variables			
	Inputs	**Outputs**	**States**	**Temporary**
	1 T: real 2 TD: real 3 u: real 4 E: logic 5 R: logic 6 IV: real	1 y: real	1 x1: real 2 x2: real	

2.10.2.3 Pseudo-code

System-Init-Code	Run-Code

x1 = 0.0;
x2 = 0.0;
	if (R) {
	    x1 = IV;
	}
	else if (E) {
	    x1 = x1 + (TD*u - TD*x2 - dT*x1)/T;
	}
	x2 = u;
	y = x1;

2.10.2.4 Test-cases

Testdata

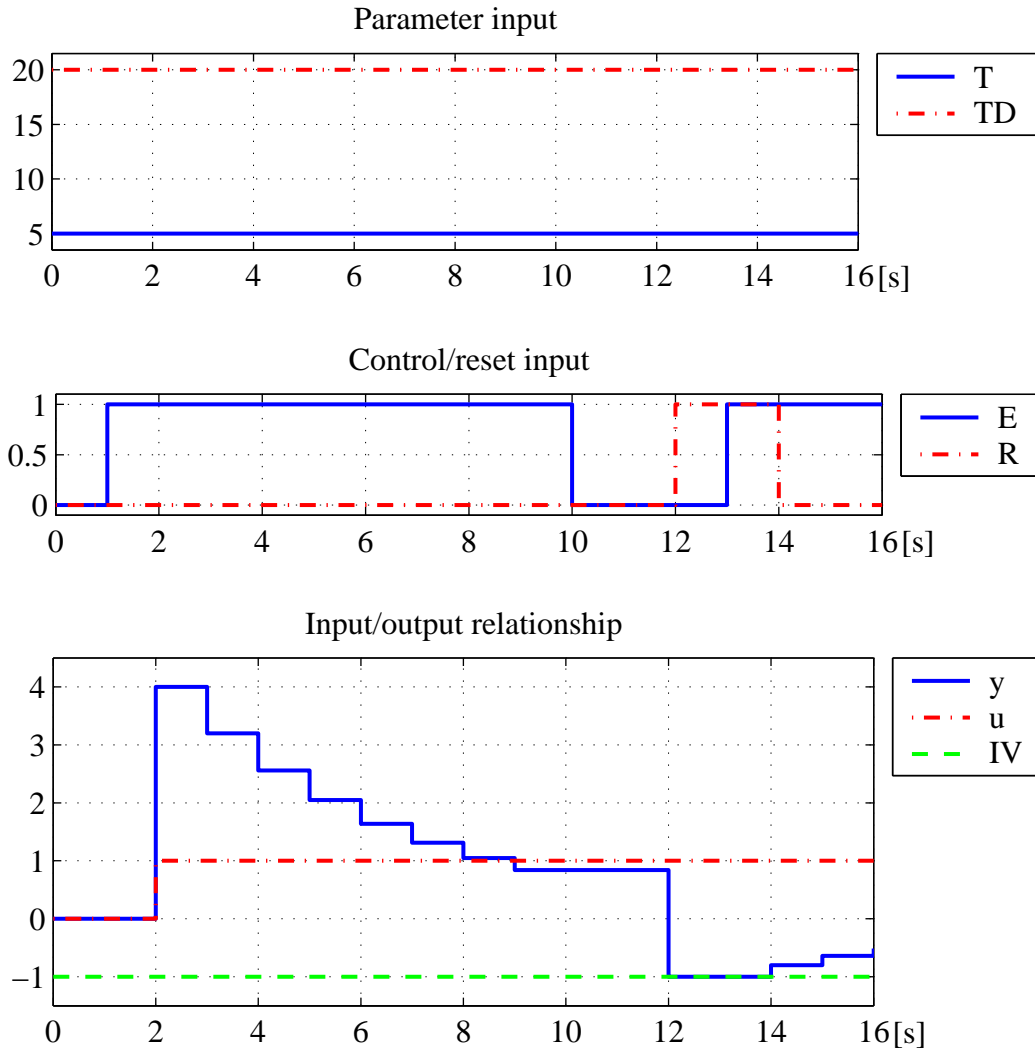```
Time vector:
t 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Input values:
T 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
TD 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
u 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
E 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
R 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
IV -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Output/state values:
x1 0 0 4 3.2 2.56 2.048 1.6384 1.3107 1.0486 0.83886 0.83886 0.83886 -1.0
-1.0 -0.8 -0.64 -0.512
x2 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
y 0 0 4 3.2 2.56 2.048 1.6384 1.3107 1.0486 0.83886 0.83886 0.83886 -1.0
-1.0 -0.8 -0.64 -0.512
```

**Simulation result**



## 2.10.3    LowpassKResetEnabled (LOWPASSK_RE)
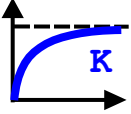
### 2.10.3.1    Verbal description

A discrete time first order lowpass filter with time constant $1/K$. Laplace notation is

$$G(s) = \frac{K}{s + K}. \tag{2.29}$$

The relationship between the analogue cut-off frequency ($f_c$) and $K$ is

$$f_c = \frac{K}{2 \cdot \pi}. \tag{2.30}$$

### 2.10.3.2 Icon and variables

| Icon | Variables | | | |
|---|---|---|---|---|
| | Inputs | Outputs | States | Temporary |
|  | 1 K: real<br>2 u: real<br>3 E: logic<br>4 R: logic<br>5 IV: real | 1 y: real | 1 x: real | |

### 2.10.3.3 Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x = 0.0; | if (R) {<br>    x = IV;<br>}<br>else if (E) {<br>    x = x + K*dT*(u-x);<br>}<br>y = x; |

### 2.10.3.4 Test-cases

**Testdata**

```
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
K 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
u 1 1 1 1 1 1 1 1 1
E 1 1 1 0 1 1 1 0 0
R 0 0 0 1 1 0 0 0 0
IV 0 0 0 0 0 0 0 0 0

Output/state values:
x 0.5 0.75 0.875 0 0 0.5 0.75 0.75 0.75
y 0.5 0.75 0.875 0 0 0.5 0.75 0.75 0.75
```
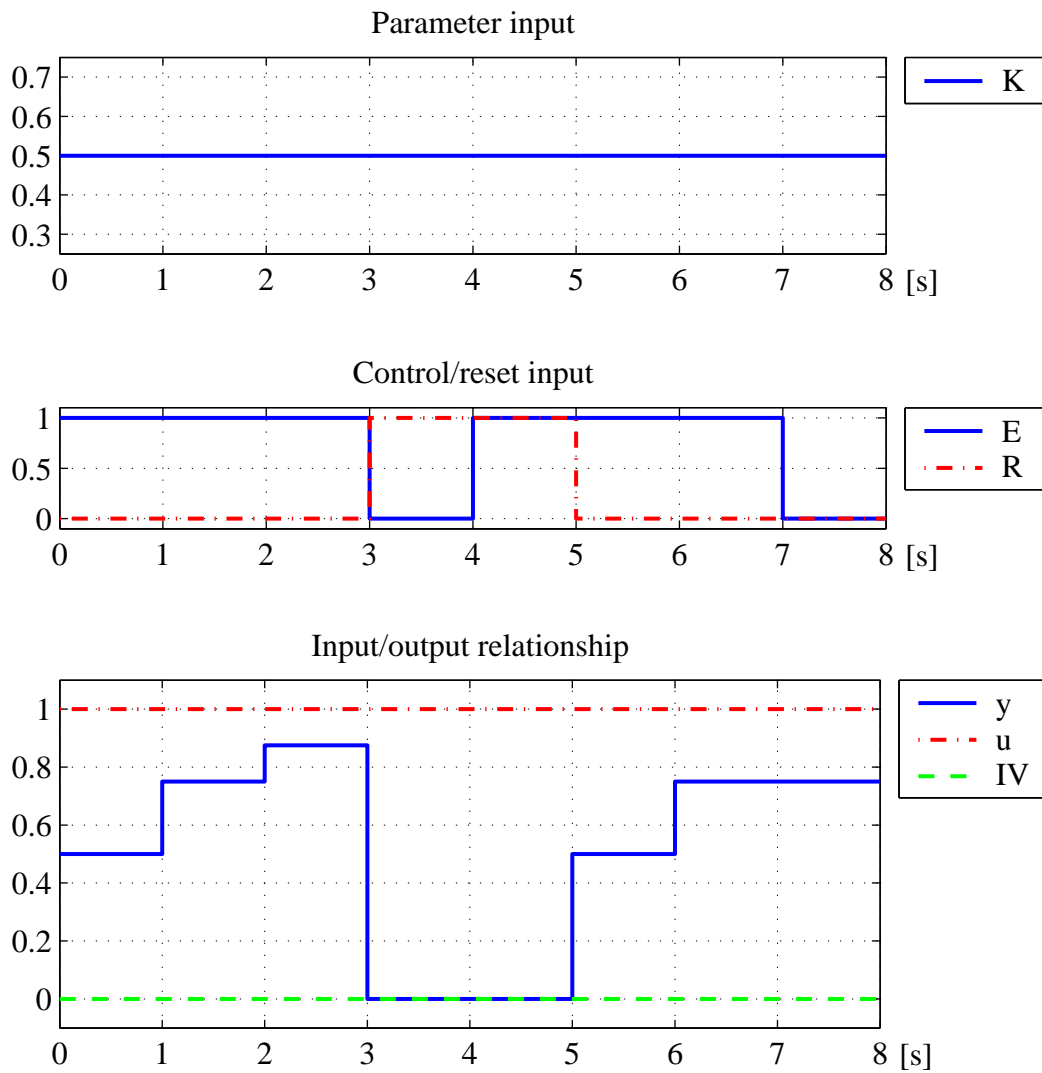
**Simulation result**

## 2.10.4   LowpassSecondOrderEnabled (LOWPASSSECORD_RE)
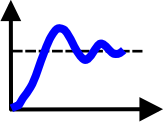
### 2.10.4.1   Verbal description

A discrete time second order low pass filter with time constant $T$ and damping factor $D$ (PT2). Laplace notation is

$$G(s) = \frac{1}{T^2 \cdot s^2 + 2 \cdot D \cdot T + 1}. \qquad (2.31)$$

The relationship between the analogue cut-off frequency $(f_c)$ and the time constant $T$ is

$$f_c = \frac{1}{2 \cdot \pi \cdot T}. \qquad (2.32)$$

### 2.10.4.2   Icon and variables

| Icon | Variables | | | |
|---|---|---|---|---|
| | **Inputs** | **Outputs** | **States** | **Temporary** |
|  | 1  T:  real<br>2  D:  real<br>3  u:  real<br>4  E:  logic<br>5  R:  logic<br>6  IV1:  real<br>7  IV2:  real | 1  y:  real | 1  x1:  real<br>2  x2:  real | 1  temp:  real<br>2  temp1:  real |

### 2.10.4.3   Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x1 = 0.0;<br>x2 = 0.0; | if (R) {<br>    x1 = IV1;<br>    x2 = IV2;<br>}<br>else if (E) {<br>    temp = x2;<br>    temp1 = pow(T,2) + 2*D*T*dT;<br>    x2 = x1;<br>    x1 = x2 + (pow(dT, 2)*(u - x2))/temp1 +<br>    pow(T,2)*(x2 - temp)/temp1;<br>}<br>y = x1; |

#### 2.10.4.4   Test-cases
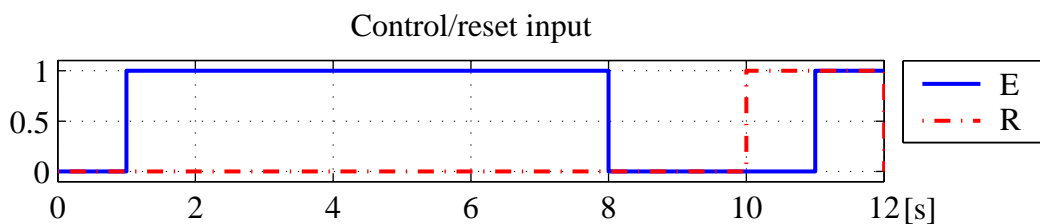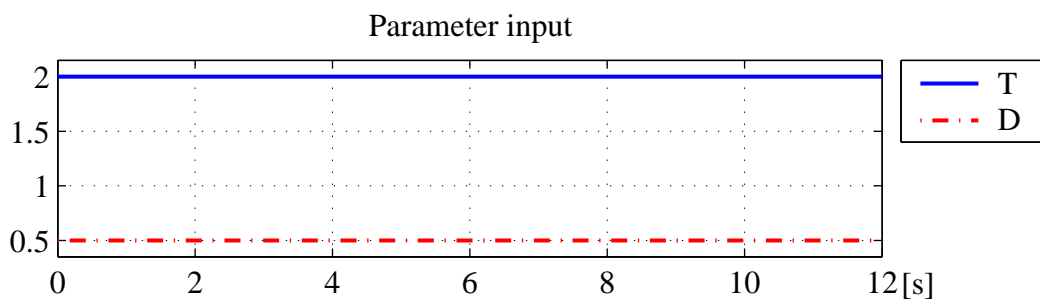
**Testdata**
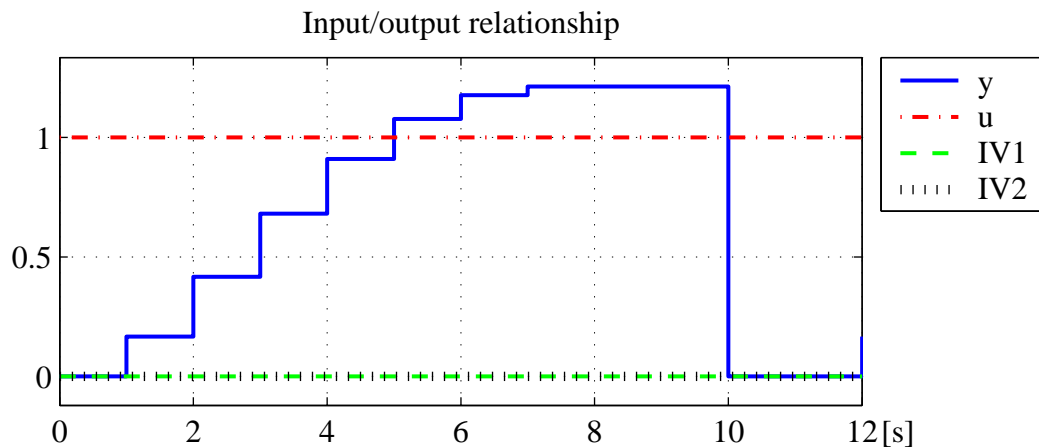
Time vector:

t 0 1 2 3 4 5 6 7 8 9 10 11 12

Input values:

T 2 2 2 2 2 2 2 2 2 2 2 2 2
D 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
u 1 1 1 1 1 1 1 1 1 1 1 1 1
E 0 1 1 1 1 1 1 1 0 0 0 1 1
R 0 0 0 0 0 0 0 0 0 0 0 1 1 0
IV1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
IV2 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Output/state values:

x1 0.0 0.16666667 0.41666667 0.68055556 0.90972222 1.0775463 1.1765046
1.2130594 1.2130594 1.2130594 0.0 0.0 0.16666667
x2 0.0 0.0 0.16666667 0.41666667 0.68055556 0.90972222 1.0775463 1.1765046
1.1765046 1.1765046 0.0 0.0 0.0
y 0.0 0.16666667 0.41666667 0.68055556 0.90972222 1.0775463 1.1765046
1.2130594 1.2130594 1.2130594 0.0 0.0 0.16666667

**Simulation result**



Parameter input



Control/reset input

Input/output relationship



## 2.10.5   LowpassTResetEnabled (LOWPASST_RE)

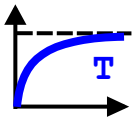### 2.10.5.1   Verbal description

A discrete time first order lowpass filter with time constant $T$. Laplace notation is

$$G(s) = \frac{1}{T \cdot s + 1}. \tag{2.33}$$

The relationship between the analogue cut-off frequency ($f_c$) and $T$ is

$$f_c = \frac{1}{2 \cdot \pi \cdot T}. \tag{2.34}$$

### 2.10.5.2   Icon and variables

| Icon | Variables | | | |
|---|---|---|---|---|
| | **Inputs** | **Outputs** | **States** | **Temporary** |
| | 1   T:   real<br>2   u:   real<br>3   E:   logic<br>4   R:   logic<br>5   IV:   real | 1   y:   real | 1   x:   real | |

### 2.10.5.3   Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| x = 0.0; | if (R) {<br>    x = IV;<br>}<br>else if (E) {<br>    x = x + dT*(u-x)/T;<br>}<br>y = x; |

### 2.10.5.4    Test-cases

**Testdata**

```
Time vector:
t 0 1 2 3 4 5 6 7 8

Input values:
T 2 2 2 2 2 2 2 2 2
u 1 1 1 1 1 1 1 1 1
E 1 1 1 0 1 1 1 0 0
R 0 0 0 1 1 0 0 0 0
IV 0 0 0 0 0 0 0 0 0

Output/state values:
x 0.5 0.75 0.875 0 0 0.5 0.75 0.75 0.75
y 0.5 0.75 0.875 0 0 0.5 0.75 0.75 0.75
```
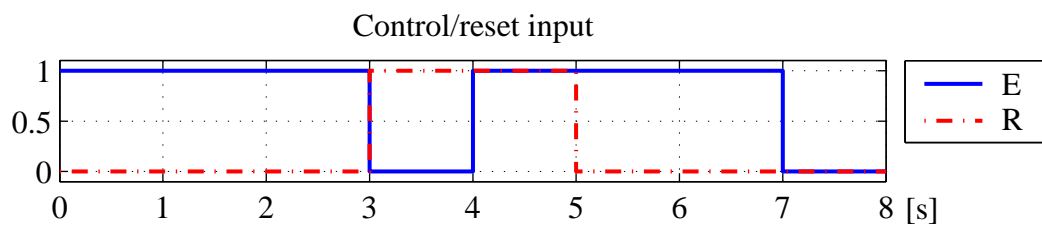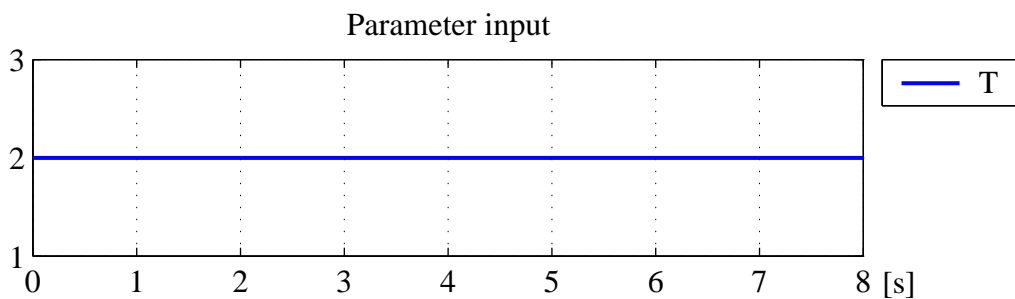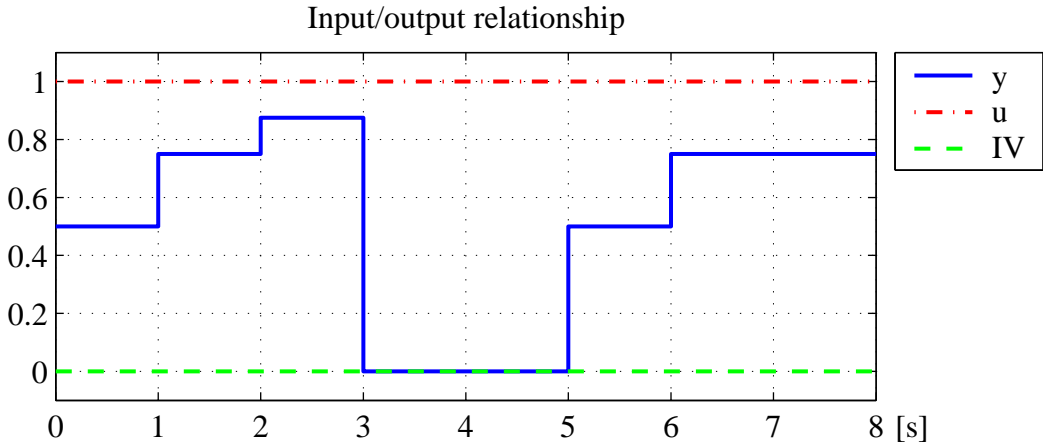
**Simulation result**



Parameter input



Control/reset input

Input/output relationship

## 2.11    Parameter and constant

### 2.11.1   Curve / 1D Index Lookup Table (PARAMETER1DINDEX)

#### 2.11.1.1   Verbal description

Performs 1D-table lookup by *index*. The algorithm searches for an interval of two x-axis values, which embrace the input value $u$. The block outputs the corresponding y-axis value of the lower x-value.
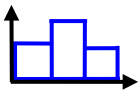
$$nx = ny = dim(P\_x) - 1; \tag{2.35}$$

$$y = \begin{cases} P\_y[0] & : & u < P\_x[0] \\ P\_y[i] & : & i := P\_x[i] < u < P\_x[i+1]; \\ & & P\_x[0] < u < P\_x[nx] \\ P\_y[nx] & : & u > P\_x[nx] \end{cases} \tag{2.36}$$

This means that the output value $y$ depends on the input value $u$ but is kept constant within a given break point interval. The function dim(x) returns the dimension of the array $x$.

Block parameter:

$P\_x$   : real    Input Vector {'numerical constant' | 'value name'}
$P\_y$   : real    Output Vector {'numerical constant' | 'value name'}

#### 2.11.1.2   Icon and variables

| Icon | Variables | | | |
|------|-----------|---|---|---|
| | Inputs | Outputs | States | Temporary |
|  | 1   u:   real | 1   y:   real | | |

### 2.11.2   Curve / 1D Interpolation Lookup Table (PARAMETER1DINTERPOLATION)

#### 2.11.2.1   Verbal description

Searches for an x-axis interval, which embraces the input value $u$. The output is the linear interpolated value in this interval. No extrapolation is performed.

$$nx = ny = dim(P\_x) - 1; \tag{2.37}$$

$$y = \begin{cases} P\_y[0] & : \quad u < P\_x[0] \\ f(u, P\_x[i], P\_x[i+1], P\_y[i], P\_y[i+1]) & : \quad i := P\_x[i] < u < P\_x[i+1]; \\ & \qquad P\_x[0] < u < P\_x[nx] \\ P\_y[nx] & : \quad u > P\_x[nx] \end{cases}$$

(2.38)

This means that the output value depends on the input value u but is interpolated within the assigned break point interval. Outside the given break point interval distribution the output value is kept constant with respect to $P\_y[0]$ or $P\_y[ny]$.

Block parameter:

$P\_x$ : real   Input Vector {'numerical constant' | 'value name'}
$P\_y$ : real   Output Vector {'numerical constant' | 'value name'}

### 2.11.2.2 Icon and variables

| Icon | Variables | | | |
| --- | --- | --- | --- | --- |
| | Inputs | Outputs | States | Temporary |
| | 1   u:   real | 1   y:   real | | |

## 2.11.3 Map / 2D Index Lookup Table (PARAMETER2DINDEX)

### 2.11.3.1 Verbal description

Performs 2-D Table lookup by index. The algorithm searches for an interval on both x-axis, which embrace the input values. Takes the index of the smaller axis-value and outputs the corresponding y-axis value.

$$\begin{aligned} nx &= dim(P\_x) - 1; \\ ny &= dim(P\_y) - 1; \\ [ny\ nx] &= dim(P\_z) - 1; \end{aligned}$$

(2.39)

$$y = \begin{cases} P\_z[0][0] & : \quad u < P\_x[0] \ \&\& \ v < P\_y[0] \\ P\_z[i][j] & : \quad P\_x[0] < u < P\_x[nx] \ \&\& \\ & \qquad P\_y[0] < v < P\_y[ny]; \\ & \qquad i := P\_x[i] < u < P\_x[i+1]; \\ & \qquad j := P\_y[j] < u < P\_y[j+1]; \\ P\_z[nx][ny] & : \quad u > P\_x[nx] \ \&\& \ v > P\_y[ny] \end{cases}$$

(2.40)

This means that the output value y depends on the input values u,v but is kept constant within a given break point interval. The function dim(x) returns the dimension of the matrix x.

Block parameter:

| | | |
|---|---|---|
| $P\_x$ | : real | Input Vector {'numerical constant' \| 'value name'} |
| $P\_y$ | : real | Input Vector {'numerical constant' \| 'value name'} |
| $P\_z$ | : real | Output Matrix {'numerical constant' \| 'value name'} |

### 2.11.3.2 Icon and variables

| Icon | Variables | | | |
|---|---|---|---|---|
| | **Inputs** | **Outputs** | **States** | **Temporary** |
|  | 1  u1:  real<br>2  u2:  real | 1  y:  real | | |

## 2.11.4 Map / 2D Interpolation Lookup Table (PARAMETER2DINTERPOLATION)

### 2.11.4.1 Verbal description

Performs 2-D linear interpolation of input values using the specified input/output table. No Extrapolation is performed.

$$
\begin{aligned}
nx &= dim(P\_x) - 1; \\
ny &= dim(P\_y) - 1; \\
[ny\ nx] &= dim(P\_z) - 1;
\end{aligned}
\tag{2.41}
$$

$$
y = \begin{cases}
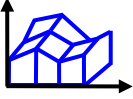P\_z[0][0] & :\ u < P\_x[0] \,\&\&\, v < P\_y[0] \\
f(u, v, P\_x[i], P\_x[i+1], P\_y[j], P\_y[j+1], & \\
\quad P\_z[i][j], P\_z[i+1][j+1]) & :\ P\_x[0] < u < P\_x[nx] \,\&\& \\
& \quad\ P\_y[0] < v < P\_y[ny]; \\
& \quad\ i := P\_x[i] < u < P\_x[i+1]; \\
& \quad\ j := P\_y[j] < u < P\_y[j+1]; \\
P\_z[nx][ny] & :\ u > P\_x[nx] \,\&\&\, v > P\_y[ny]
\end{cases}
\tag{2.42}
$$

This means that the output value y depends on the input values u,v but is interpolated within a given break point interval. Outside the given breakpoint interval distribution the output value is kept constant with respect to $P\_z$.

Block parameter:

| | | |
|---|---|---|
| $P\_x$ | : real | Input Vector {'numerical constant' \| 'value name'} |
| $P\_y$ | : real | Input Vector {'numerical constant' \| 'value name'} |
| $P\_z$ | : real | Output Matrix {'numerical constant' \| 'value name'} |

**2.11.4.2   Icon and variables**

| Icon | Variables | | | |
|------|-----------|--|--|--|
| | **Inputs** | **Outputs** | **States** | **Temporary** |
|  | 1   u1:   real<br>2   u2:   real | 1   y:   real | | |

## 2.12   Signal path switches

### 2.12.1   Switch (SWITCH)

#### 2.12.1.1   Verbal description

The boolean input $l$ decides which input is returned:

$$y(n) = \left\{ \begin{array}{lcl} u_1(n) & : & l == true \\ u_2(n) & : & l == false \end{array} \right. \qquad (2.43)$$

#### 2.12.1.2   Icon and variables

| Icon | Variables | | | |
|---|---|---|---|---|
| | Inputs | Outputs | States | Temporary |
| | 1  l:    logic<br>2  u1:  real<br>3  u2:  real | 1  y:  real | | |

#### 2.12.1.3   Pseudo-code

| System-Init-Code | Run-Code |
|---|---|
| | if (l) {<br>    y = u1;<br>}<br>else {<br>    y = u2;<br>} |

#### 2.12.1.4   Test-cases

**Testdata**
```
Time vector:
t 0 1 2 3

Input values:
l 0 1 1 0
u1 1 1 1 1
u2 2 2 2 2
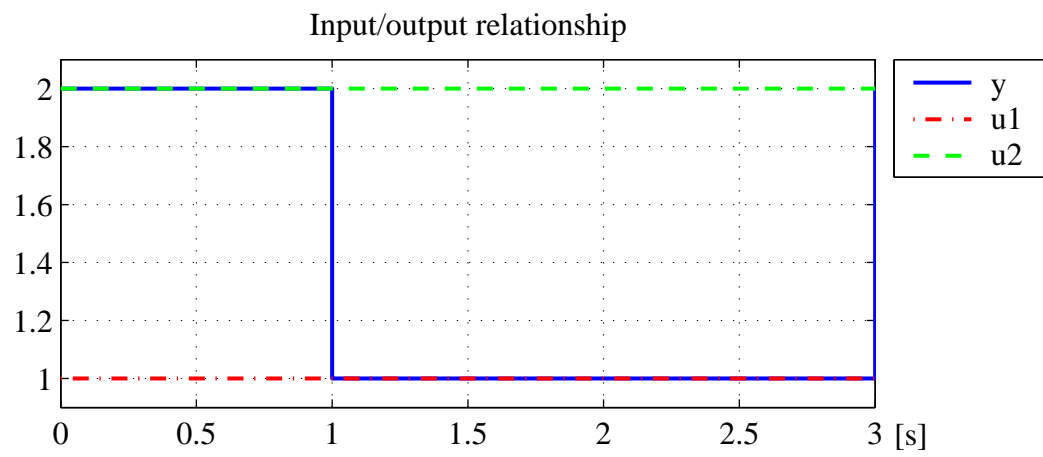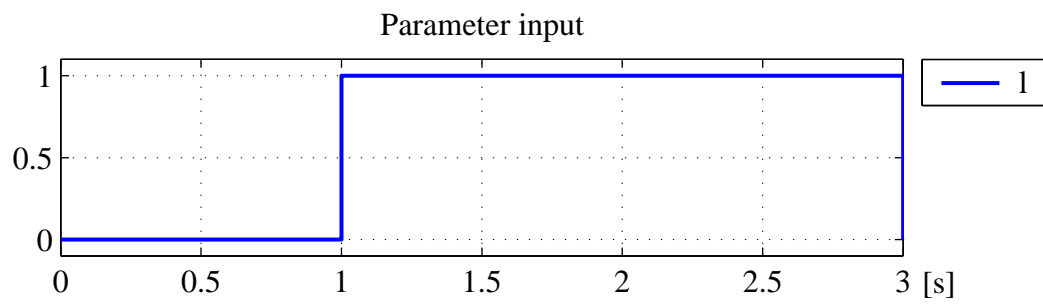
Output/state values:
y 2 1 1 2
```

**Simulation result**

Parameter input

Input/output relationship

# Kapitel 3

# Glossar

| Variable | Signal Type | Description |
|---|---|---|
| B_min | Boolean Output Signal | minimum saturation active flag |
| B_max | Boolean Output Signal | maximum saturation active flag |
| dT | System Parameter | sample time |
| D | Parameter Input | damping of second order lowpass |
| IV | Parameter Input | initial value |
| IV1 | Parameter Input | initial value 1 |
| IV2 | Parameter Input | initial value 2 |
| K | Parameter Input | lowpass gain excluding dT |
| LD | Parameter Input | limitation down |
| LSP | Parameter Input | left switching point |
| LU | Parameter Input | limitation up |
| m | Parameter Input | lowpass gain including dT |
| MN | Parameter Input | minimum limitation |
| MX | Parameter Input | maximum limitation |
| n | Parameter Input | number of samples |
| P_x | Parameter Constant | input vector of curve |
| P_y | Parameter Constant | output/input vector of curve/map |
| P_z | Parameter Constant | output vector of map |
| R | Boolean Input Signal | Reset |
| RSP | Parameter Input | Hysteresis Right Switching Point |
| RT | Boolean Input Signal | Reset Trigger Pulse |
| T | Parameter Signal | Delay Time or Time Constant |
| TD | Parameter Signal | Highpass Time Constant |
| u | Input Signal | standard input variable |
| u1 | Input Signal 1 | standard input variable |
| u2 | Input Signal 2 | standard input variable |
| UMAX | Parameter Signal | maximum input limit |
| UMIN | Parameter Signal | minimum input limit |

| Variable | Signal Type | Description |
|----------|-------------|-------------|
| v | Input Signal | only used in 2-D-Look-Up Table |
| y | Output Signal | standard output variable |