Function Sheet

Extent shown:	
Blocks chosen:	ABK, APP, FB, FDEF, QUER
System:	SG-ME9-0
Project:	MEDC17
Project code:	GEA2SI/P040C700;0
Responsible:	Gabriele Eilber
Department:	GS-EC/EDG1
Phone:	33717
Date of issue:	06.AUG.2007

Program release: P040C700 (Predecessor :)

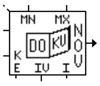
Table of contents: Sections

FU ASCETBLK 1.10.0 Description of ASCET block library

FDEF ASCETBLK 1.10.0 Function definition

Representation of engine management functions:

To represent engine management functions physical information (data flow)
is distinguished from digital control information (control flow).
Data flow: load signal, speed, control factor
Control flow: condition idle, switch driving position, error catalyst
Solid lines mark the data flow, dotted lines mark the control flow.

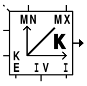


Baseblocks (general description):
- Blocks with marker "NOV" store the value of the output (integrator value, flag, RAM-cell) in a non volatile RAM (otherwise in a volatile RAM). The behaviour of these blocks is identical to their pendants without "NOV" marker.
- The main inputs and outputs ("in" and "out") have no symbol in the block icon; if there are no other characteristics their default values are 0.0 (float) bzw. FALSE (bool).
- Unconnected inputs are configured with 0.0 (float) bzw. FALSE (bool), if there are no other agreements.
- Some blocks have a special input at their left upper corner (default TRUE), usually connected with a calculation raster trigger to determine the frequency of calculation explicitly. Raster time describes the time gap between two cycles.
- Any deviation from following default configuration of input and output values will be described in the documentation of each block.

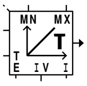
	breve in icon	default value	designation
INPUTS:	E	TRUE	enable calculation
	I	FALSE	initialization
	IV	0.0	initial value
	K	0.0	here: integration factor K
	MX	1E35	upper limitation of output
	MN	-1E35	lower limitation of output

ascetblk-teil0gb

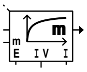
ascetblk-teil0gb



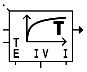
Integrator K
new value of integrator := old value of integrator + K * rasterTime * in
INPUTS: K integration factor



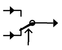
Integrator T
new value of integrator := old value of integrator + (rasterTime / T) * in
The minimum value of T is limited to rasterTime.
INPUTS: T time constant of integration



Recursion
new value := old value + m * (in - old value)
INPUTS: m weighting factor

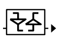


Lowpass
new value of lowpass := old value of lowpass + (rasterTime / T) * (in - old value of lowpass)
The minimum value of T will be limited to rasterTime.
INPUTS: T time constant

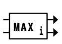


Input-Switchover
The icon shows the default position of the switch, default value of non-wired inputs is 0.0 .

Exclusive-OR
Output-signal is TRUE when exactly one input-signal is TRUE.



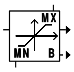
Edge (bidirectional)
A positive or negative edge at input triggers a TRUE at output during this simulation step.



Maximum (2 inputs)
The value of output is equal to maximum of all input values.
Output i indicates the index of first input generating maximum.

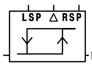
ascetblk-teil1gb

ascetblk-teil1gb



Limiter
The value of output-signal is equal to input-signal limited to the range [MN, MX].
TRUE at output "B" indicates an active limitation, othetherwise this output is FALSE.

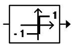
Absolute value
The value of output-signal is equal to absolute value of input-signal.



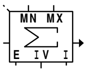
Hysteresis
Wiring of hysteresis determines right and left breakover point of hysteresis:

wired	left breakover point	right breakover point
LSP und delta	LSP	LSP + delta
LSP und RSP	LSP	RSP
delta und RSP	RSP - delta	RSP

All other wirings of the inputs LSP, RSP and delta produce FALSE as output-value (incorrect wiring).



Signum
Input value < 0.0, causes -1.0 at output, otherwise output value is equal to 1.0 .



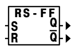
Accumulator
The accumulator-value is changed by input-value and limited to the range [MN, MX].

FLAG
Imitation of one volatile 1 bit-RAM-cell.

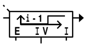
RAM
Imitation of an volatile RAM-cell.

ascetblk-teil2gb

ascetblk-teil2gb



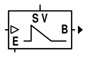
RS-FlipFlop
The RS-FlipFlop has a set-input S and a reset-input R, reset dominates set.
The value of output !Q is always inverted to output Q.



Delay raster
Signal delay by one cycle of calculation grid, out(i) := in(i-1).
A non-wired raster-input causes signal delay of one simulation step.

Turn-off delay
If input-signal switches from TRUE to FALSE the output-signal follows this alteration with a delay. Delay time is determined by input DELAY. Switching input to TRUE during delay time causes TRUE at output immediately.

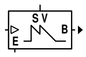
Turn-on delay
If input-signal switches from FALSE to TRUE the output-signal follows this alteration with a delay. Delay time is determined by input DELAY. Switching input to FALSE during delay time causes FALSE at output immediately.



Timer
A positive Edge at input starts the timer running down:
- the start value of timer is determined by value at input SV (in seconds).
- the output switches to TRUE and stays TRUE until timer expires.

Further positive edges at input have no effect until the timer time has expired.
FALSE at input E stops the timer until TRUE at this input causes continuation of running down.

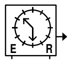
INPUTS:	in	start timers
	SV	start value of timer
OUTPUTS:	B	timer is running



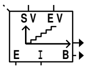
Timer-Retrigger
Basic function like "Timer", difference: A further positive edge at input causes always restart of timer.

ascetblk-teil3gb

ascetblk-teil3gb

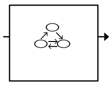


Time-Counter
TRUE at input resets time-counter to 0.0. R = FALSE, starts time-counter.
FALSE at input E stops the time-counter. The time-counter output shows the elapsed time in seconds.
INPUTS: R reset of time-counter



Counter
The counter will be incremented or decremented in every simulation step. True at input I causes determination of count-direction depending on actual startvalue and endvalue.
If value of input SV is greater than value of input EV then counter counts up, otherwise it counts down until endvalue is reached. Reaching endvalue will be indicated by TRUE at output B.
The counter may be stopped wih help of input E.

INPUTS:	SV	startvalue of counter
	EV	endvalue of counter
	I	initialize and start counter
OUTPUTS:	B	endvalue reached



Statemachine
The control flow is represented by logical gate arrays and statemachines. The sequence control in a statemachine may be modeled with help of states and transitions.
State: (ellipsis) In a statemachine is always exactly one state active, that means all corresponding actions will be done. The name of every state is represented inside the ellipsis.

Transition: (arrow) The Transition from one state to an other one in case of fulfilling the state transition condition. All corresponding transition actions will be done.
The condition, that will be checked for transition, is shown beside the transition arrow.
The string beside the transition is only a logical name representing the condition. The afterfollowing text contains the ful description. The condition with the lowest number has the highest priority.
Every statemachine has exactly one welldefined start state (S) and a reset state (R).
If RESET-condition is fulfilled in any state, the reset state becomes actual state for next simulation step.

ascetblk-teil4gb

ascetblk-teil4gb

ABK ASCETBLK 1.10.0 Abbreviations

FB ASCETBLK 1.10.0 Function description

APP ASCETBLK 1.10.0 Application hints

FU ASCETSDB 1.25.1 ASCET-SD descripton of block library

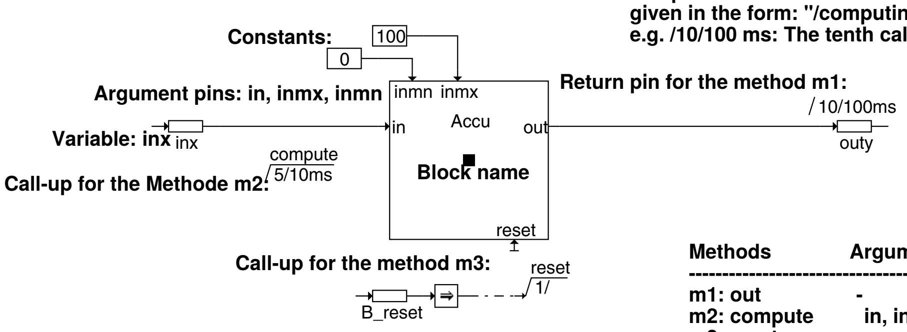
FDEF ASCETSDB 1.25.1 Function definition

Graphical presentation of basic elements

Basic elements are presented in the diagram as rectangular blocks. Communication between basic elements is displayed by connecting lines. The interfaces between basic elements are the pins at the block edges. Each block has return pin that outputs the result from the block. In addition to this, there are argument pins that provide the inputs into the block as well as method pins that are used for those methods without input arguments and without return values.

The methods call up functions in the block.

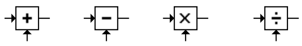
The process information and the computing sequence are given in the form: "/computing sequence/process", e.g. /10/100 ms: The tenth call-up in the 10-ms computing frame



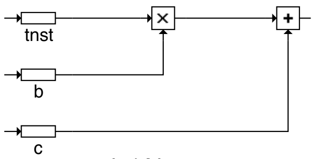
Methods	Arguments	Return value
m1: out	-	Float
m2: compute	in, inmx, inmn	-
m3: reset	-	-

The example given above shows a block with 3 methods:
- The method m1 "out" has one return value
The method "out" is called up by the request for the return value from the subsequent block outy, that is in the tenth position in the computing sequence in the 100-ms computing frame.
- The method m2 "compute" has three arguments (in, inmn, inmx) yet no return value.
The method "compute" is called up at the fifth position in the computing sequence in the 10-ms computing frame.
- The method m3 "reset" has neither arguments nor a return value. This is therefore represented by the "method pin".
If B_reset is true, then the method "reset" is called up first (1/) in the computing sequence.

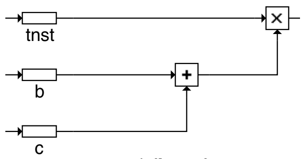
Arithmetic operations



Equations can be described using arithmetic operations (addition, subtraction, multiplication and division). Equations are represented graphically such that the return value of an operation is the argument for the subsequent operation.

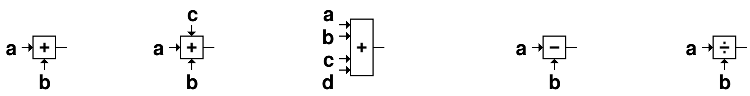


(a * b) + c

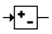


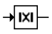
a * (b + c)

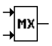
The arguments of primitive operations and their computing sequence are shown in the following:

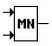


a + b	a + b + c	a+b+c+d	a - b	a / b

a  b **Negation: b = - a**

a  b **Amount: b = |a|**

a, b  c **Maximum for input values: c = MAX(a,b)**

a, b  c **Minimum for input values: c = MIN(a,b)**

Variables

receive_message

Receive message are input variables of the function that are made available from another function.

send_receive_message

Send/Receive variables are output variables of the function that are used both within as well as outside of the function.

send_message

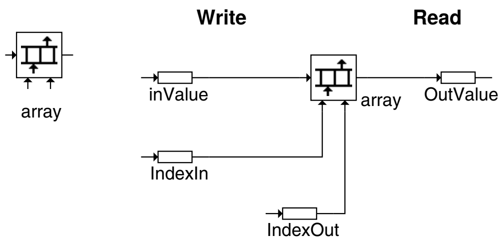
Send messages are output variables of the function and are available for the other functions.

local_varable

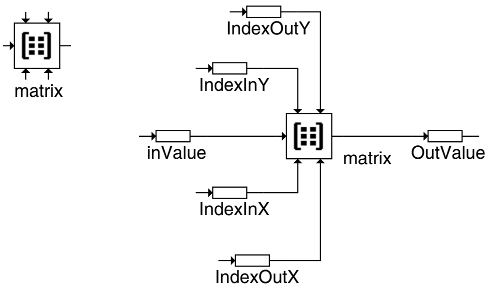
Local variables are only made available and used within the function.

Arrays and matrices

Arrays and matrices have two methods for reading and writing access to the elements. The reading and writing can be independent of one another.



Array:
- The value to be written is connected to the left pin, the associated index to the lower left pin.
- The value to be read is connected to the right pin, the associated index to the lower right pin.

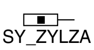


Matrix:
Matrices behave as arrays, whereby the methods here have two index arguments (x, y):
- The index x is connected to the lower left, the index y to the upper left pin for writing access.
- The index x is connected to the lower right, the index y to the upper right pin for reading access.

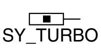
1 -1.3 **Constants**

true false **Boolean Constants**

System constants

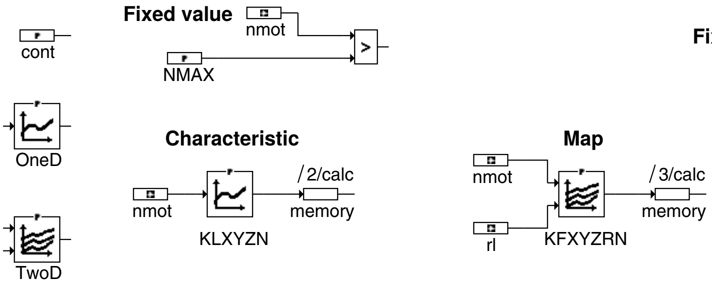

SY_ZYLZA

System constants are constants that are permanently anchored in the program and cannot be applied. System constants can switch functional positions conditionally on or off.


SY_TURBO

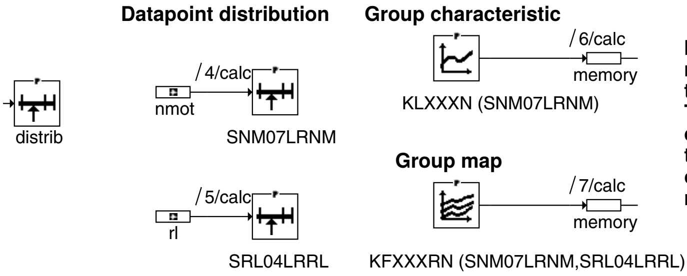
Example **SY_ZYLZA: Cylinder number**
SY_TURBO: Engine with or without turbo-charger

Fixed values, Characteristics, Maps, Group characteristics, Group maps and datapoint distribution



Fixed values are applicable parameters

Characteristics have one argument.
Maps have two arguments as the input.
Both have one return value



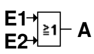
In the case of group characteristics and group maps, several characteristics and maps pick up the same datapoint distributions. To this end, the current datapoint from the datapoint distribution, e.g. SNM07LRNM, is first computed from the dependent parameter e.g. nmot. The computation of the output value for the group characteristic or map then follows using this current datapoint.

ascetsdb-a6-kl-kf-g-en

Bit operations

AND element:

E1	E2	A
0	0	0
1	0	0
0	1	0
1	1	1

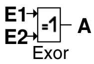


OR element:

E1	E2	A
0	0	0
1	0	1
0	1	1
1	1	1

Negation

E	A
0	1
1	0



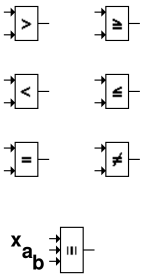
Exklusive OR:

E1	E2	A
0	0	0
1	0	1
0	1	1
1	1	0

ascetsdb-a7-bitoper-en

Comparator

The comparator provides TRUE at the output if the comparison applies.
If the comparison is not fulfilled, then FALSE is given as the output.

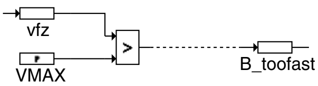


Greater, greater than or equal to

The comparison is always read from top to bottom (interval excepted):

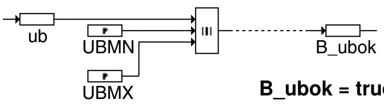
Less, less than or equal to

If vfz is greater than VMAX, then the condition B_toofast is TRUE



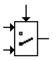
Equal, unequal

Closed interval:
a <= x <= b



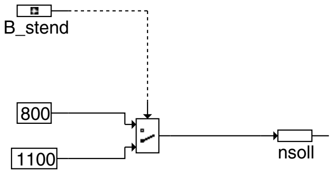
B_ubok = true if UBMN <= ub <= UBMX

ascetsdb-a8-verglei-en



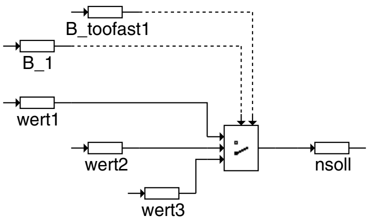
Multiplex operator "Multiplexer", "Switch"

**A multiplexer switches a value through to the output as a dependency of the input conditions.
The multiplexer icon is shown in the dormant position, i.e. if the input conditions are false.**



Example "Simple multiplexer"
- if B_stend = false: nsoll = 1100
- if B_stend = true: nsoll = 800

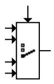
**An input condition is assigned to each value in the case of cascaded multiplexers.
The uppermost value, the input condition for which is true, is switched through.
If there is no input condition of true, then the lowest value is switched through.**

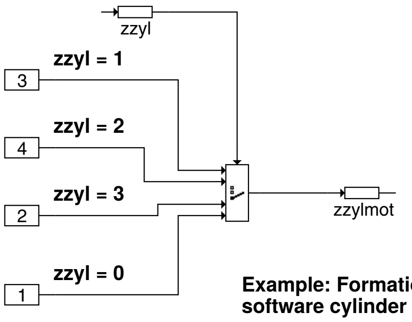


Example "Mehrfach-Muxer":
- if B_1 = true: nsoll = value1
- if B_1 = false & B_2 = true: nsoll = value2
- if B_1 = false & B_2 = false: nsoll = value3

ascetsdb-a9-multipl-en

CASE operator





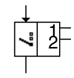
**The CASE operator switches one of the remaining left inputs through to the output as a dependency of a top-applied discrete control value (1, 2, 3, ...).
If the control value 1 is the first value, then 2 is the second value and is switched through immediately.
If the value is outside of the range, then the lowest input (default) is switched through.**

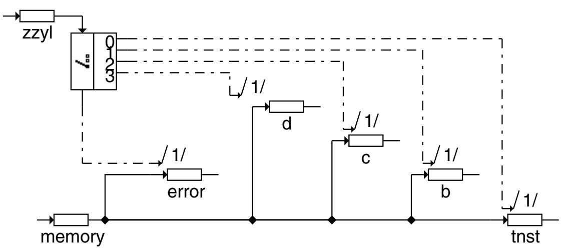
Example: Formation of the physical cylinder number onto the software cylinder number.

ascetsdb-a10-case-o-en

Switch

**The SWITCH operator activates the matching control flows over the right-hand outputs as a dependency of an upper applied discrete control value (1, 2, 3, ...).
If there is no matching output existing, then the control flow is activated at the lower output.**

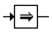




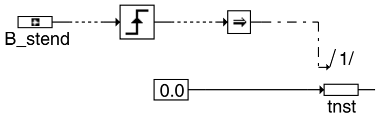
Example:
One of the following operations is executed depending on zzyl:
- if zzyl = 0: a = memory
- if zzyl = 1: b = memory
- if zzyl = 2: c = memory
- if zzyl = 3: d = memory
- otherwise: error = memory

ascetsdb-a11-kontro-en

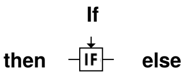
If .... then



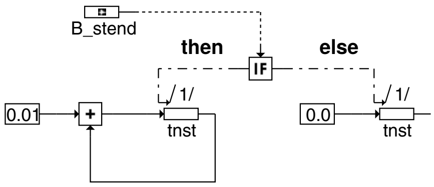
The If .. Then operation analyzes a logic condition and activates all computing sequences for TRUE that are connected to the control flow. The computing sequence is defined by the numbering.



Example:
If B_stend changes to true, then tnst = 0 is set.



The IF .. Then .. Else operation analyzes a logic condition and activates all computing sequences of the then control branch for TRUE, and all computing sequences of the else control branch for FALSE. The computing sequence at each control branch is defined by the numbering.



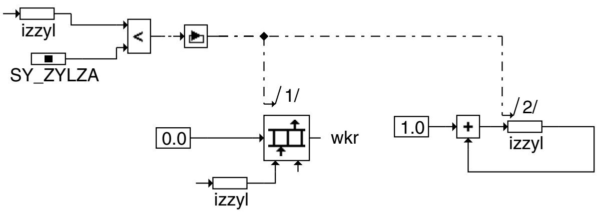
Example: If B_stend = true,
then tnst is incremented by 0.01 sec.
in the 10-ms time frame.
Otherwise (B_stend = false) tnst = 0 is set.

Mule bridge for IF:

I=True <-- IF --> F=False

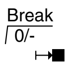
While loop

The computing sequence within the control flow of the loop is executed for such a time as the input condition is fulfilled, i.e. is TRUE. The loop is aborted when the input condition is FALSE. The value for terminating the While loop is normally formed within the loop. This usually concerns a counter here that shall count up to a certain value.



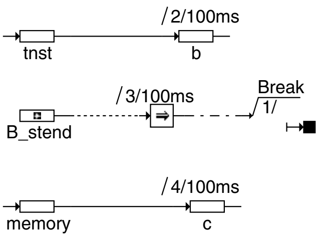
Example:
The array wkr[i] is written with 0 for such a time as izzy < SY_ZYLZA. Each element of the array is initialized with 0 by the numerical variable izzyl at the index input of the array.

Break



The break operation prematurely interrupts a process, e.g. the functions component in a computing frame. All subsequent calculations of the function in the process with a higher number for the computing sequence are not then executed.

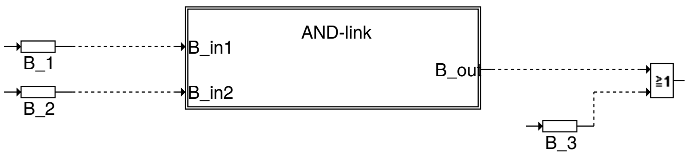
Example:



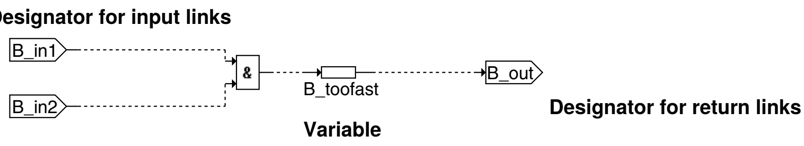
In accordance with the computing sequence, a break is triggered after the operation b = a exactly then when B_stend = TRUE. The 100-ms process is started if a break occurs. The subsequent operation, c = memory, is no longer executed.

Hierarchy:

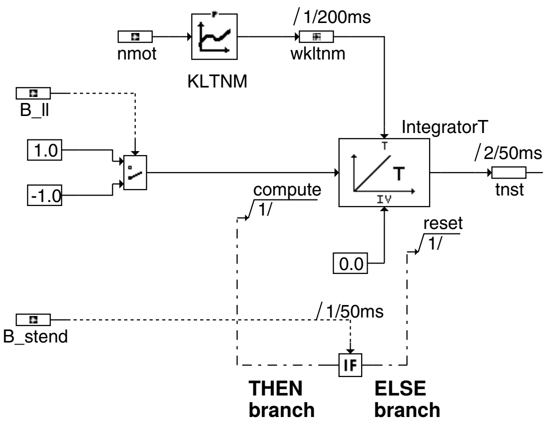
The hierarchy is graphical possibility for structuring functions.
The hierarchy block is identified by a double border.
The corresponding hierarchy level is identified by the name, here "AND link".
The transfer element is only a designator for the links between both levels.



Function in the graphical "AND link" hierarchy:



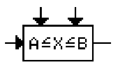
Example



The method "reset" of the integrator is executed in the 50-ms computing frame as long as the condition B_stend = FALSE. This method causes the internal memory cell of the integrator to initialized with the IV-value, i.e. 0.0. If the condition now becomes B_stend = TRUE, then the left-hand control flow is activated and the method "compute" has as the arguments time T and the input value. This depends on B_ll. The input value = 1.0 for B_ll = TRUE; the input value = -1.0 for B_ll = FALSE. The number in front of the computing frame indicates the computing sequence: The time constant T is computed in the 200-ms computing frame and the message wkltnm is stored in Send/Receive. The IF .. THEN .. ELSE query is first executed in the 50-ms computing frame. The integration value is written into the variable a in the second step.

ASCET-SD System Library

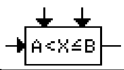
Comparators



ClosedInterval

`ClosedInterval` returns TRUE, if the value `x` is in the closed interval defined by A and B.

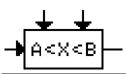
Methods	Characteristics	Arguments	Return Value
out	TRUE is returned, if `A <= x <= B`. Otherwise FALSE is returned.	x::continuous A::continuous B::continuous	TRUE or FALSE



LeftOpenInterval

`LeftOpenInterval` returns TRUE, if the value `x` is in the left open interval defined by A and B.

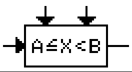
Methods	Characteristics	Arguments	Return Value
out	TRUE is returned `A < x <= B. Otherwise FALSE is returned.`	x::continuous A::continuous B::continuous	TRUE or FALSE



OpenInterval

`OpenInterval` returnes TRUE, if the value `x` is in the open interval defined by A and B .

Methods	Characteristics	Arguments	Return Value
out	TRUEis returned, if `A < x < B`. Otherwise FALSE is returned .	x::continuous A::continuous B::continuous	TRUE or FALSE



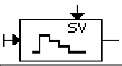
RightOpenInterval
`RightOpenInterval` returnes TRUE, if the value x is in the right open interval defined by A and B.

Methods	Characteristics	Arguments	Return Value
out	TRUE is returned, if `A <= x < B`.Otherwise FALSE is returned.	x::continuous A::continuous B::continuous	TRUE or FALSE

GreaterZero
`GreaterZero` returns TRUE, if the value x is greater than zero.

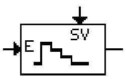
Methods	Characteristics	Arguments	Return Value
out	TRUE is returned, if `x > 0.0`. otherwise FALSE is returned.	x:continuous	TRUE or FALSE

Counter & Timer



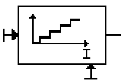
CountDown
CountDown decrements the counter and signals when the counter has reached zero.

Methods	Characteristics	Arguments	Return Value
start	The counter is set to the start value .	startValue::unsigned discrete	none
compute	The counter is decremented by one.	none	none
out	TRUE is returned if the counter is greater than zero. Otherwise FALSE is returned.	none	TRUE or FALSE



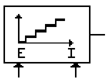
CountDownEnabled
CountDownEnabled dercrements the counter and signals when the counter has reached zero. This counter must be enabled explicitly.

Methods	Characteristics	Arguments	Return Value
start	The counter is set to start value.	startValue:: unsigned discrete	none
compute	If enable isTRUE, the counter is decremented by one.	enable::TRUE or FALSE	none
out	TRUE is returned if the counter is greater zero.Otherwise FALSE is returned.	none	TRUE or FALSE



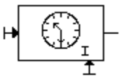
Counter
Counter increments the couner by one.

Methods	Characteristics	Arguments	Return Value
reset	The counter is set to zero.	none	none
compute	The counter is incremented by one .	none	none
out	The counter value is returned .	none	unsigned discrete



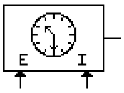
CounterEnabled
CounterEnabled increments the counter by one. This counter must be enabled explicitly.

Methods	Characteristics	Arguments	Return Value
reset	If `initEnable is` TRUE, the counter is set to zero .	initEnable::TRUE or FALSE	none
compute	If `enable` is TRUE, the counter is incremented by one.	enable::TRUE orFALSE	none
out	The counter value is returned.	none	unsigned discrete



StopWatch
StopWatch increments the time counter by one dT.

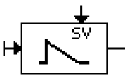
Methods	Characteristics	Arguments	Return Value
reset	The counter is set to zero.	none	none
compute	The time counter is inremented by dT.	none	none
out	The time counter value, i. e. the time elapsed since the last start, is returned .	none	continuous



StopWatchEnabled
StopWatchEnabled increments the time counter by one dT. Thies timer must be enabled explicitly.

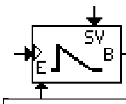
Methods	Characteristics	Arguments	Return Value
reset	If `initEnable is` TRUE, the internal counter is set to zero.	initEnable::TRUE or FALSE	none
compute	If `enable` is TRUE, the time counter is incremented by dT .	enable::TRUE or FALSE	none
out	The time counter value, i.e. the time elapsed since the last start and while `enable` was TRUE is returned.	none	continuous

ascetsdb-etassys4-en



Timer
Timer decrements the time counter by dT and singals when the time counter has reached zero . Its not retriggerable.

Methods	Characteristics	Arguments	Return Value
start	The time counter is set to zero, if the time counter value was previously less than or equal to zero.	startTime::continuous	none
compute	The time counter is decremented by dT.	none	none
out	TRUE is returned, if the time counter is greater then zero. Otherwise FALSE is returned.	none	continuous



TimerEnabled
TimerEnabled decrements the time counter by dT and signals when the time counter reaches zero. It must be enabled explicitly.

Methods	Characteristics	Arguments	Return Value
compute	If `enable` is TRUE, `in` has a rising edge and the time counter value is less or equal zero, the timer is started, i.e. its counter value is set to the start time. Otherwise the time counter is decremented by dT. If `enable` is FALSE nothing happens.	enable::TRUE or FALSE in::TRUE or FALSE startTime::continuous	none
out	TRUE is returned, if the time counter value is greater than zero. Otherwise , FALSE is returned .	none	continuous

ascetsdb-etassys5-en