Chapter 8

CONDITION

Edition 3

Overview

Contents (Continued)

Contents (Continued)

Contents (Continued)

Overview

This function block class contains a range of general purpose signal conditioning function blocks. A random number generator is also included in this class.

SCALE FUNCTION BLOCK

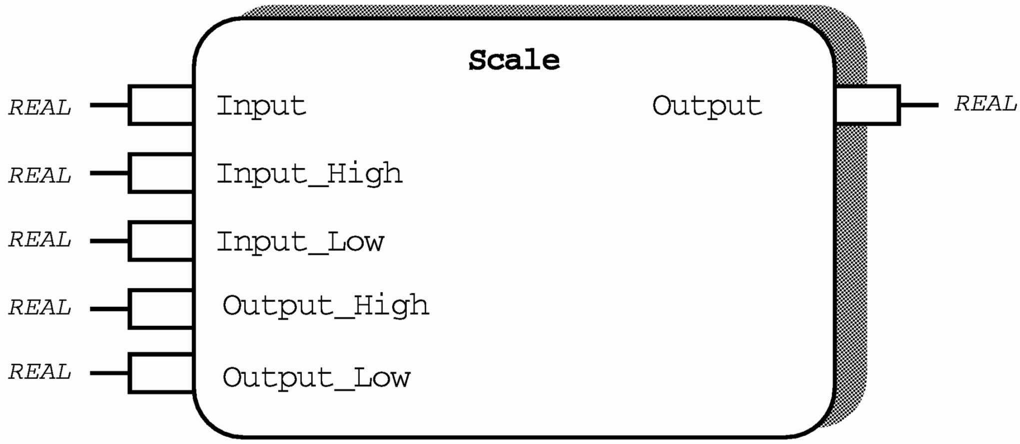


Figure 8-1 Scale Function Block Diagram

Functional Description

This function block enables a value to be re-scaled by means of an input span and an output span. The high and low input values with their corresponding high and low output values are set. The function block will then provide an output which is the same percentage of the output span as the input value is of the input span. Put mathematically:

$$\text{Scale\_fb.Output} = \text{Output\_Low} + \text{Scale\_fb.Input} * \frac{(\text{Output\_High} - \text{Output\_Low})}{(\text{Input\_High} - \text{Input\_Low})}$$

There is no overange system built into the block, so if an input value goes outside the input span (i.e. greater than **Input_High** or lower than **Input_Low**) then the same linear relationship applies.

Function Block Attributes

Type: 1C01

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Input, Output

Memory Requirements: ....24 Bytes

Parameter Descriptions

Input (IN)

The value that is to be scaled.

Input_High (IHI)

The upper value of the Input. This value, if applied as Input, would give an Output equal to **Output_High.**

Input_Low (ILO)

The lower value of the input. This value, if applied as Input, would give an Output equal to **Output_Low.**

Output_High (OHI)

The upper value of the Output matching **Input_High**. It is equal to the Output that would be obtained when a value of **Input_High** is applied as Input.

Output_Low (OLO)

The lower value of the Output matching **Input_Low**. It is equal to the Output that would be obtained when a value of **Input_Low** is applied as input.

Output

The scaled equivalent of Input.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Input	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Input_High	**REAL**	100	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Input_Low	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Output_High	**REAL**	10	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Output_Low	**REAL**	-10	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Output	**REAL**	-10	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38

Table 8-1 Scale Parameter Attributes

MAXIMUM/MINIMUM/AVERAGER FUNCTION BLOCK

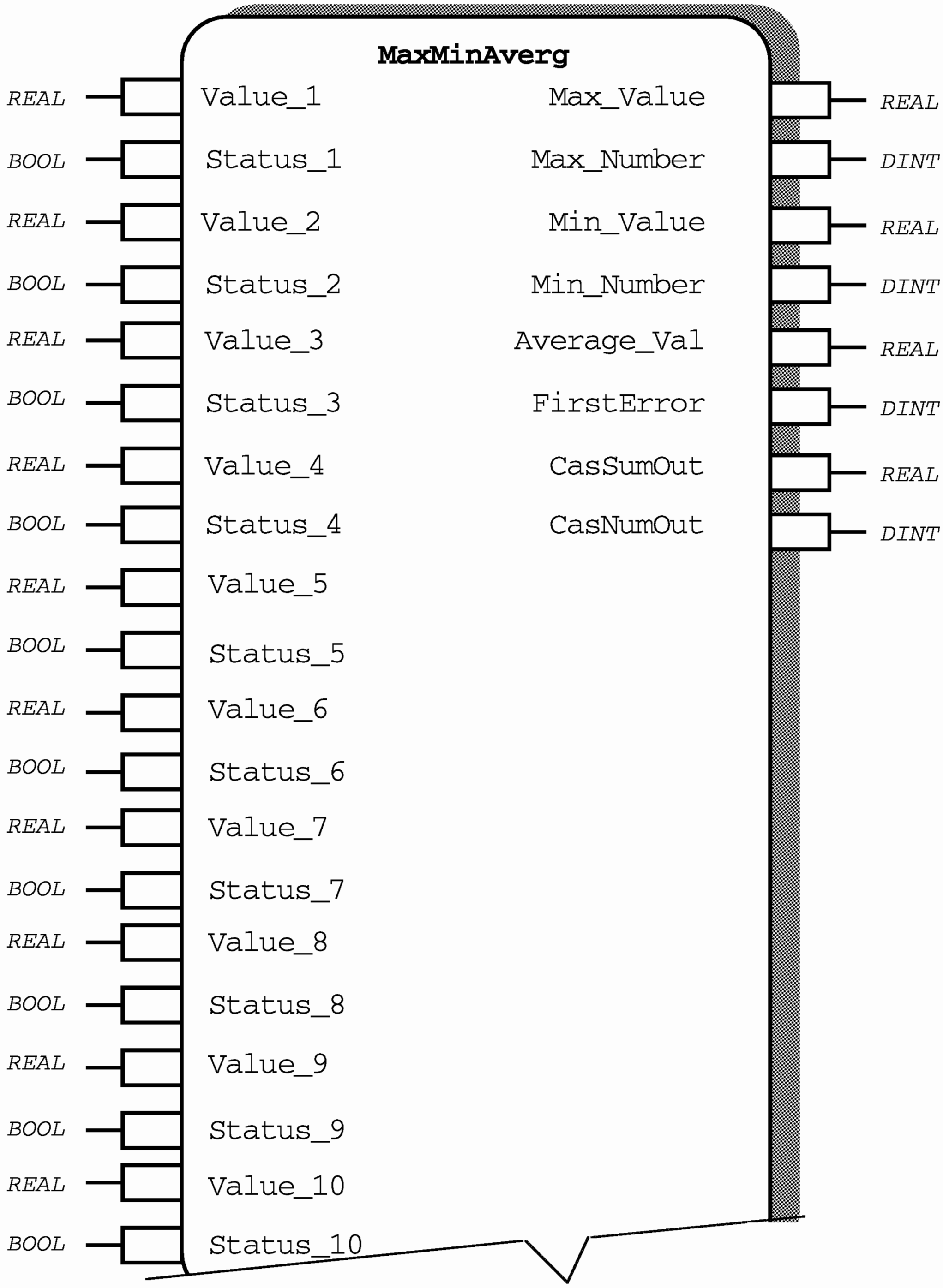


Figure 8-2 MaxMinAverg Function Block Diagram

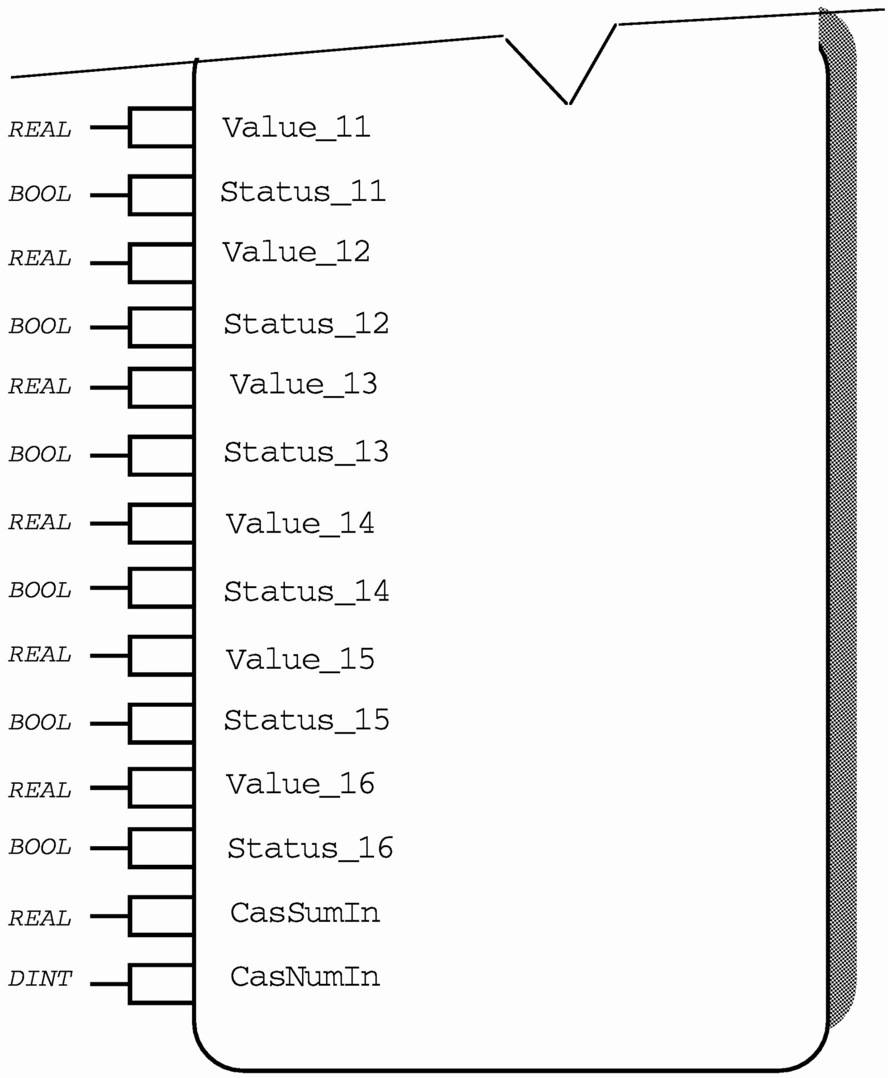


Figure 8-2 MaxMinAverg Function Block Diagram (continued)

Functional Description

The maximum, minimum and average of up to sixteen inputs is derived and these calculated values are presented as outputs. The inputs are numbered one to sixteen, and the number of the input which carries the minimum value and the number of the input which carries the maximum value are given as two further outputs.

Outputs are also included to allow the sum and number of inputs to be cascaded through a series of blocks so that the overall average of the inputs to all blocks can be calculated.

Inputs may be individually enabled or disabled, and the number of the first disabled channel is reported on an output.

As an example of the uses to which this function block might be put, consider the situation where a number of thermocouple inputs are to be used to control the temperature of a large workpiece. The thermocouples would be wired to the inputs of the Maximum/Minimum/Averager function block, and the average temperature could be used as the Process Variable to a PID block. The maximum and minimum outputs could be used as alarms, or, in the situation where a ramp function block is being used, to activate hold-back. The status output of each of the analog input function blocks could be used to enable the relevant input to the Maximum/Minimum/Averager function block. The FirstError output would then indicate the lowest-numbered input on which there was a hardware fault.

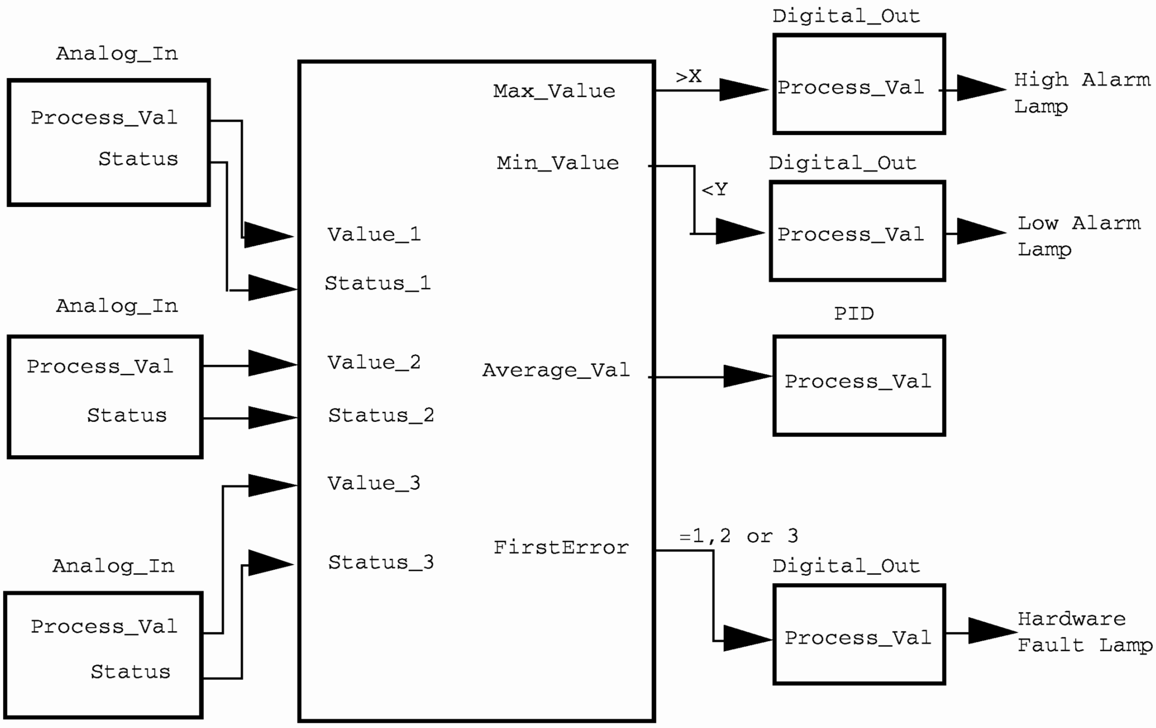


Figure 8-3 Example of using MaxMinAverg Function Block to provide input to PID Loop.

Function Block Attributes

Type: 1C08
Class: ....CONDITION
Default Task: ....Task_2
Short List: ....Max_Value, Min_Value, Average_Val,
....First_Error.
Memory Requirements: ..... 220 Bytes

Parameter Descriptions

Value_1 (V1) to Value_16 (V16)

The inputs to the function block which are to be processed.

Status_1 (S1) to Status_16 (S16)

Allows each Value input to be individually enabled or disabled. **Status_1** enables **Value_1**, **Status_2** enables **Value_2** etc.

CasSumIn (CSI)

Used to bring in the sum from a previous block so that the average calculated is the average of the inputs to this block and the inputs to a previous block or previous blocks. Used in conjunction with CasNumIn.

CasNumIn (CNI)

Used to bring in the number of enabled inputs from a previous block so that the average calculated is the average of the inputs to this block and the inputs to a previous block or previous blocks. Used in conjunction with CasSumIn.

Max_Value (MXV)

The highest value currently applied to any of the enabled inputs.

Max_Number (MXN)

The number of the input which currently has the highest value applied to it. For example, if the highest value of any enabled input is 27, and this is being applied to the input Value_8, then **Max_Number** will be 8 whilst **Max_Value** will be 27.

Min_Value (MNV)

The lowest value currently applied to any of the enabled inputs.

Min_Number (MNN)

The number of the input which currently has the lowest value applied to it. For example, if the lowest value of any enabled input is 20, and this is being applied to the input Value_11, then **Min_Number** will be 11 whilst **Min_Value** will be 20.

Average_Val (AV)

The arithmetic average value of all enabled inputs to this block, if CasNumIn and CasSumIn are zero:

$$\text{Average\_Val} = \frac{\Sigma\ (\text{enabled inputs})}{(\text{number of enabled inputs})}$$

However, if CasNumIn is positive, and CasSumIn is non-zero, then Average_Val is the average of the enabled inputs to this and all previous blocks cascaded via CasNumOut/CasNumIn and CasSumOut/CasSumIn.For the current block this might be expressed as:

$$\text{Average\_Val} = \frac{\Sigma\ (\text{enabled inputs}) + \text{CasSumIn}}{(\text{number of enabled inputs}) + \text{CasNumIn}}$$

Imagine two blocks cascaded together. If ten values are enabled on the first block whose sum is 428, **Average_Val** will be 42.8, CasNumOut will be 10 and CasSumOut will be 428. The two cascade outputs of the first block will be wired to the two cascade inputs on the second block (first.CasSumOut to second.CasSumIn and first.CasNumOut to second.CasNumIn). On this second block a further six Values are enabled whose sum is 270. **Average_Val** will be 43.63 ((270+428)/(10+6)), whilst CasNumOut will be 16 and CasSumOut will be 698.

FirstError (FER)

The number of the first input which is disabled will be indicated here. For example, if all inputs are enabled except **Value_12** and **Value_16**, FirstError will be set to 12.

CasSumOut (CSO)

The sum of all enabled inputs and CasSumIn.

CasNumOut (CNO)

The sum of the number of enabled inputs and CasNumIn.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Value_1 to Value_16	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Status_1 to Status_16	**BOOL**	Disable (0)	Oper	Oper	Senses	Disable (0) Enable (1)
CasSumIn	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
CasNumIn	**DINT**	0	Oper	Oper	High Limit Low Limit	2147483647 0
Max_Value	**REAL**	-3·4E + 38	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Max_Number	**DINT**	0	Oper	Block	High Limit Low Limit	16 0
Min_Value	**REAL**	3·4E + 38	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Min_Number	**DINT**	0	Oper	Block	High Limit Low Limit	16 0
Average_Value	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
FirstError	**DINT**	1	Oper	Block	High Limit Low Limit	16 0
CasSumOut	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
CasNumOut	DINT	0	Oper	Block	High Limit Low Limit	2147483647 0

Table 8-2 Maximum/Minimum Average Parameter Attributes

INPUT SWITCHOVER FUNCTION BLOCK

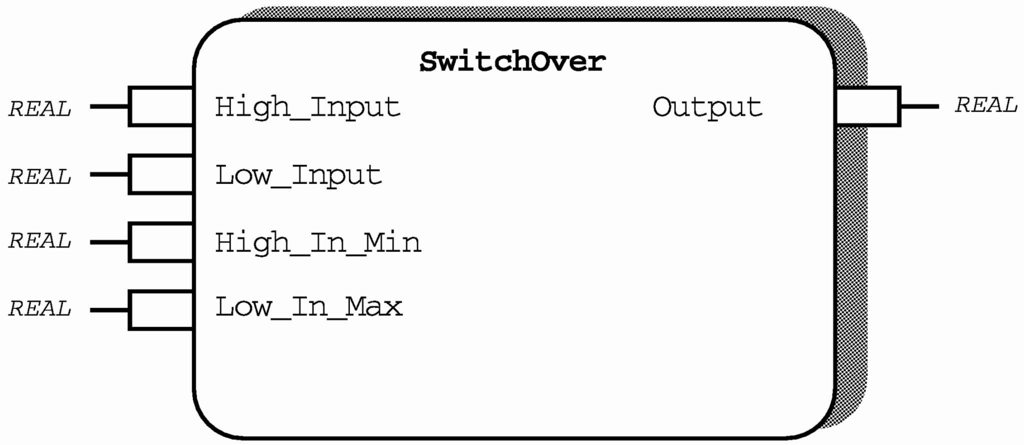


Figure 8-4 Input Switchover Function Block Diagram

Functional Description

When measurements are to be made over a wide range it is sometimes the case that more than one transducer needs to be used, one for the lower part of the range and another for the upper end. In the middle of the range there is a range of values where an aggregate of the two values is used; this is known as the overlap range. This sort of arrangement is used for example in temperature measurements using thermocouple and pyrometer, and for vacuum systems using high- and rough-vacuum gauges.

This function block has two inputs which can be wired to the **Process_Vals** of the input function blocks handling the higher scale input transducer (**High_Input)** and the lower scale input transducer (**Low_Input**). Two other inputs to the function block define the maximum value of validity for the lower scale input (**Low_in_Max**) and the minimum value of validity for the higher scale input (**High_In_Min**).

Note: This block did not function correctly in Firmware 3.00 - upgrade to 3.12

The Output of the function block comes from **High_Input** whilst this is above the overlap range, or the **Low_Input** when this is below the overlap range. When the two inputs are within the overlap range, the output is the arithmetic average of the inputs.

Function Block Attributes

Type: 1C10

Class: .... CONDITION

Default Task: .... Task_2

Short List: .... High_Input, Low_Input, Output.

Memory Requirements ...... 28 Bytes

Parameter Descriptions

High_Input (HI)

The value from a transducer representing the upper part of the scale, i.e. values above **High_In_Min.** would normally be wired to the **Process_Val** of an **Analog_In** function block.

Low_Input (LI)

The value from a transducer representing the lower part of the scale, i.e.values below **Low_In_Max**. would normally be wired to the **Process_Val** of an **Analog_In** function block.

High_In_Min (HMN)

The value below which values of **High_In** will be disregarded.

Low_In_Max (LMX)

The value above which values of **Low_In** will be disregarded.

Output (OP)

Whilst the **High_Input** is above **Low_In_Max**, Output comes exclusively from **High_Input** . When **High_Input** drops below **Low_in_Max**, Output is either from **Low_Input** (if **Low_Input** is below **High_In_Min)** or the arithmetic average of **High_Input** and **Low_Input** (if **Low_Input** is between **High_In_Min** and **Low_In_Max**).

In the unlikely situation of **High_Input** being below **High_in_Min** and **Low_Input** being above **Low_In_Max**, Output will be the arithmetic average of **High_Input** and **Low_Input** .

Low Input	<High_In_Min	High_Input <High_In_Min and <Low_In_Max	<Low_In_Max
<High_In_Min	*Low_Input*	*Low_Input*	*High_Input*
<High_In_Min and <Low_In_Max	*x(Low_Input) + (1-x)High_In_Min*	*x(Low_Input) + (1-x) High_Input*	*High_Input*
<Low_In_Max	$\frac{Low\_Input + High\_Input}{2}$	*x(Low_In_Max) + (1-x) High_Input*	*High_Input*

Where $x = \frac{(Low\_In\_Max-Old\_Output)}{(Low\_In\_Max-High\_In\_Min)}$

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
High_Input	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Low_Input	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
High_In_Min	**REAL**	500	Oper	Oper	High Limit Low Limit	+3·402823E+38 Low_In_Max
Low_In_Max	**REAL**	1000	Oper	Oper	High Limit Low Limit	High_In_Min -3·402823E+38
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38

Table 8-3 SwitchOver Parameter Attributes

PROCESS DELAY FUNCTION BLOCK

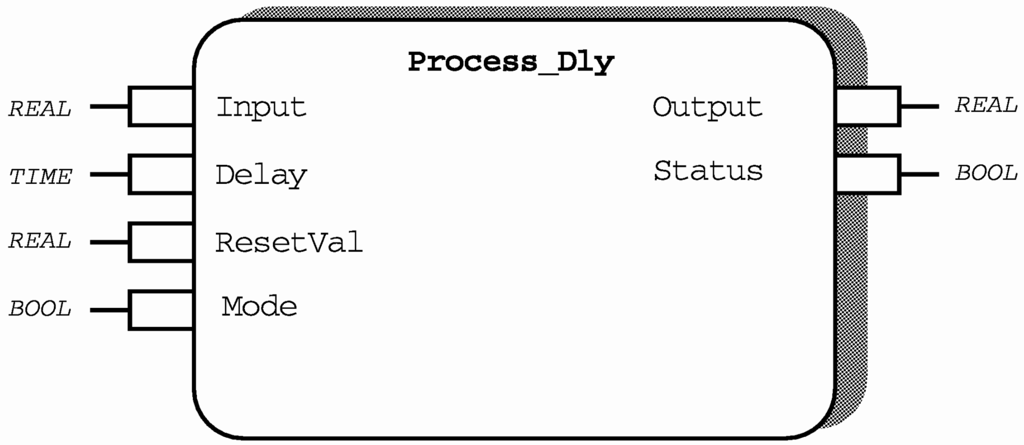
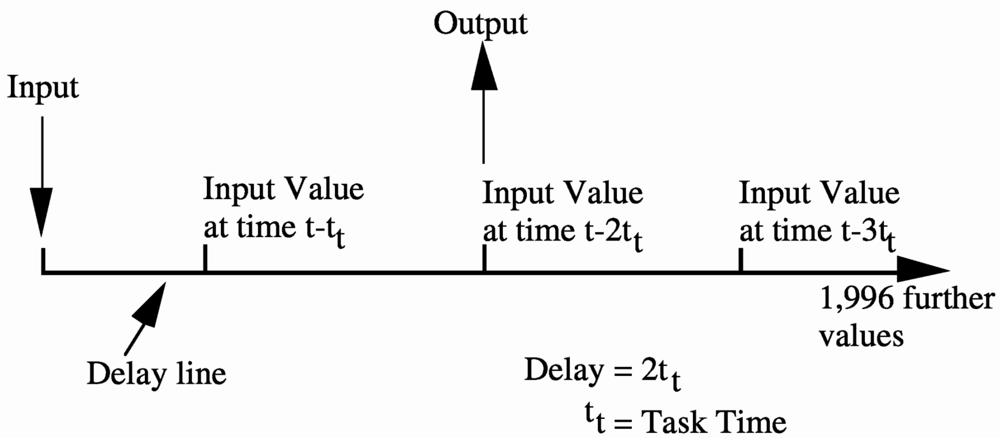


Figure 8-5 Process Delay Function Block Diagram

Functional Description

A "delay line" is used to store the value of the Input of this function block as a function of time. The Output displays the value that was on the Input of the function block a specified period ago (provided the Mode is set to Run).

A change to the Input parameter thus passes down this delay line and after a specified time the Output displays this changed input value.



The values on the delay line shift one place along every time the function block is executed so with default tasking every 100ms they all move along. The current value is snapshotted onto the first position on the delay line after all the moves are made. Since there are 2000 positions on the delay line, the maximum delay time is

3 minutes 2seconds with default tasking. Longer delays can be achieved by running the function block at a slower task rate.

All values on the entire line can be reset to a chosen value set via ResetVal and by setting Mode to Reset.

Because the delay line is a constant time-length, if the Delay is increased it is possible to see the same change occuring on the output that has already been seen. This is analogous to being passed by a motor car as you sit by the roadside, then being magically transported half a mile up the road and watching it drive past once more.

Function Block Attributes

Type: 1C20

Class: .... CONDITION

Default Task: .... Task_2

Short List: .... Input, Output, Delay, Status.

Memory Requirements: ..... 8042 Bytes

Parameter Descriptions

Input (IN)

The value to be put on the delay line.

Delay (TD)

The interval required between a change occuring on the Input and the same change appearing on the Output. It may also be considered as the time distance that the Output is situated down the delay line.

ResetVal (R)

All values on the delay line will be set to this value when the Mode is set to Reset.

Mode (M)

Switchable between Run and Reset, this input allows the delay line to be updated from the Input (Run mode) or all values on the delay line to be set to ResetVal (Reset mode).

Output (OP)

The value that was on the Input at a time equal to Delay ago. If Delay is more than the maximum value allowed (2000 * task time), the output will show the last value on the delay line, i.e the maximally delayed value.

Status (S)

Normally this output will show a Go status. However, if Delay is more than the maximum value allowed (2000 * task time), the Status will show TooLong, indicating that the requested delay is not being achieved.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Input	**REAL**	0	Oper	Oper	High Limit Low Limit	+3402823E+38 -3·402823E+38
Delay	**TIME**	10s	Oper	Oper	High Limit Low Limit	Depends on task time 0ms
ResetVal	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Mode	**BOOL**	Run(0)	Oper	Oper	Senses	Run(0) Reset (1)
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Status	**BOOL**	Go(0)	Oper	Block	Senses	Go (0) TooLong (1)

Table 8-4 Process Delay Parameter Attributes

HYSTERESIS REAL FUNCTION BLOCK

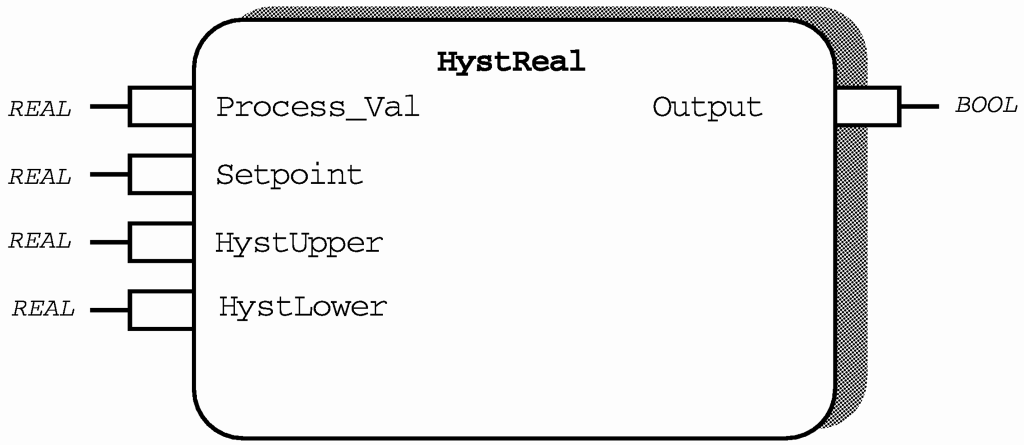


Figure 8-6 Hystersis Real Function Block Diagram

Functional Description

When monitoring analogue values there are many situations where some form of hysteresis is required. This function block allows the independent setting of an upper and lower hysteresis point on either side of a setpoint for values of type Real. When the **Process_Val** goes above Setpoint plus the upper hysteresis limit, HystUpper, the Output switches on and stays on until **Process_Val** goes below Setpoint minus the lower hysteresis limit, HystLower.

By adjusting the upper and lower hysteresis values it is possible to create a number of different strategies for causing the output to activate or de-activate. By setting HystUpper to equal **Setpoint,** classic high alarm hysteresis can be created: **Output** becomes true when **Setpoint** is exceeded, and only goes false again when the **Process_Val** has fallen lower than **Setpoint** minus HystLower.

By setting HystLower to zero, and inverting the **Output** in subsequent wiring statements, classic low alarm hysteresis can be created: the **Output** becomes false when **Process_Val** goes below setpoint, and only goes true again when the input has risen higher than **Setpoint** plus HystUpper. Other possibilities arise by using HystUpper and HystLower in tandem.

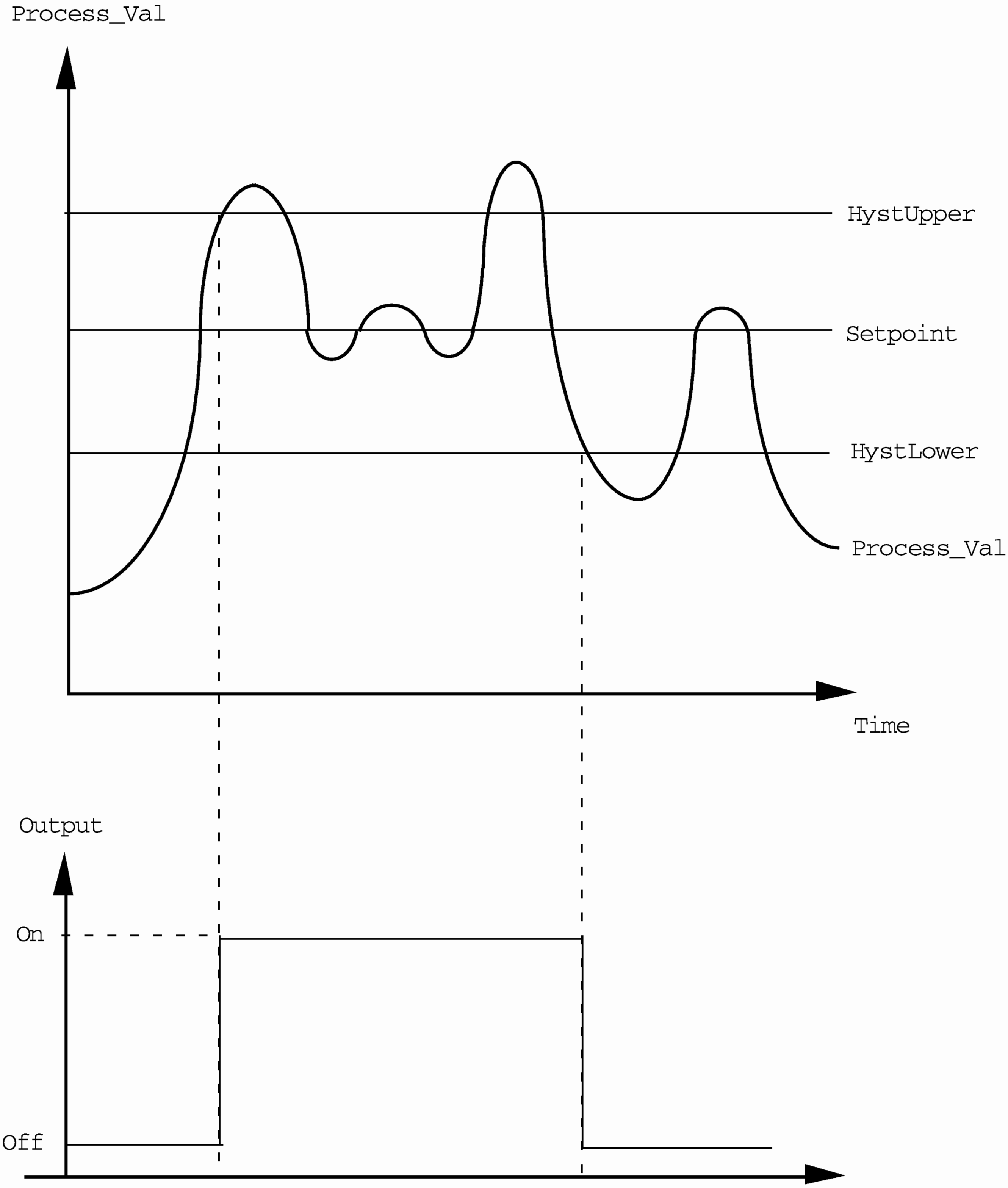


Figure 8-7 Upper and Lower Hysteresis

Function Block Attributes

Type:...1C60
Class: ....CONDITION
Default Task: ....Task_2
Short List: ....Process_Val, Setpoint, Output
Memory Requirements: ....18 Bytes

Parameter Descriptions

Process_Val (PV)

The value to be compared against the **Setpoint.** The variable on which an alarm is required would be wired to this input.

Setpoint (SP)

The principal value against which **Process_Val** is to be compared.

HystUpper (HYU)

The positive offset from setpoint which is the level at which **Output** will switch on. Hysteresis values are relative to setpoint.

HystLower (HYL)

The negative offset from setpoint which is the level at which **Output** will switch off. Hysteresis values are relative to setpoint.

Output (OP)

This output will be on when **Process_Val** is greater than **Setpoint** plus HystUpper, off when **Process_Val** is less than **Setpoint** minus HystLower, and will retain its current value when between **Setpoint** plus HystUpper and **Setpoint** minus HystLower**.**

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Proces_Val	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Setpoint	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
HystUpper	**REAL**	1	Oper	Oper	High Limit Low Limit	+3·402823E+38 0
HystLower	**REAL**	1	Oper	Oper	High Limit Low Limit	+3·402823E+38 0
Output	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)

Table 8-5 Hystersis Real Parameter Attributes

HYSTERESIS DINT FUNCTION BLOCK

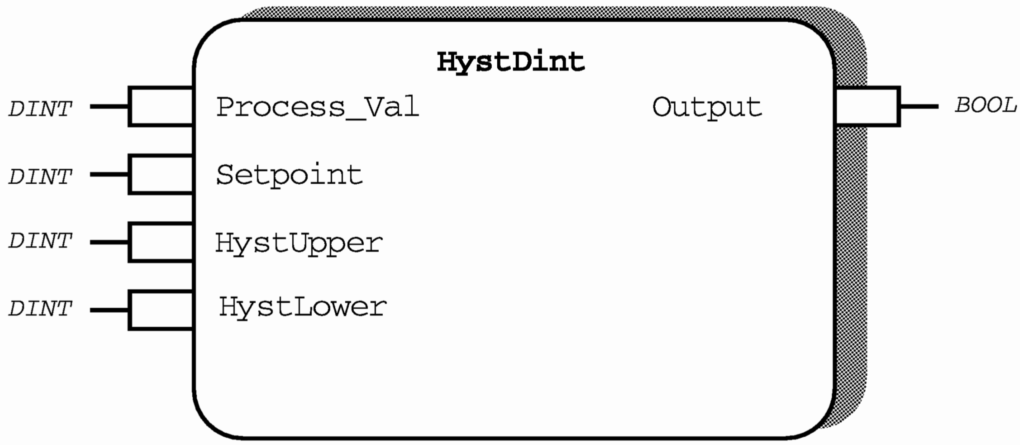


Figure 8-8 Hystersis Dint Function Block Diagram

Functional Description

When monitoring analogue values there are many situations where some form of hysteresis is required. This function block allows the independent setting of an upper and lower hysteresis point on either side of a setpoint for values of type Dint. When the **Process_Val** goes above **Setpoint** plus the upper hysteresis limit, HystUpper, the **Output** switches on and stays on until **Process_Val** goes below **Setpoint** minus the lower hysteresis limit, HystLower.

By adjusting the upper and lower hysteresis values it is possible to create a number of different strategies for causing the output to activate or de-activate. By setting HystUpper to equal **Setpoint,** classic high alarm hysteresis can be created: Output becomes true when **Setpoint** is exceeded, and only goes false again when the **Process_Val** has fallen lower than **Setpoint** minus HystLower.

By setting HystLower to zero, and inverting the **Output** in subsequent wiring statements, classic low alarm hysteresis can be created: the **Output** becomes false when **Process_Val** goes below setpoint, and only goes true again when the input has risen higher than **Setpoint** plus HystUpper. Other possibilities arise by using HystUpper and HystLower in tandem.

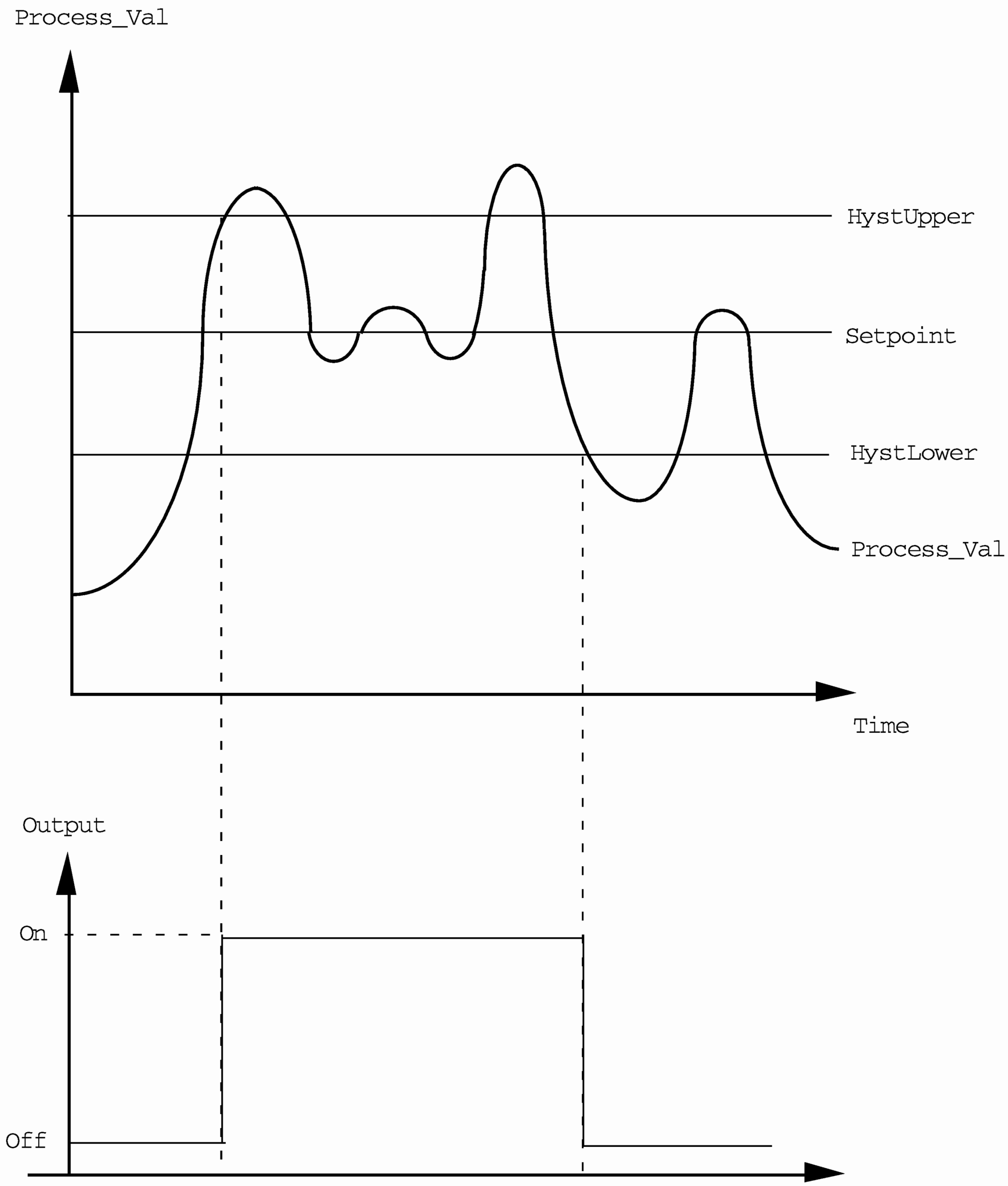


Figure 8-9 Upper and Lower Hysteresis

Function Block Attributes

Type: 1C62
Class: ....CONDITION
Default Task: ....Task_2
Short List: ....Process_Val, Setpoint, Output
Memory Requirements: ....18 Bytes

Parameter Descriptions

Process_Val (PV)

The value to be compared against the **Setpoint.** The variable on which an alarm is required would be wired to this input.

Setpoint (SP)

The principal value against which **Process_Val** is to be compared.

HystUpper (HYU)

The positive offset from setpoint which is the level at which **Output** will switch on. Hysteresis values are relative to setpoint.

HystLower (HYL)

The negative offset from setpoint which is the level at which **Output** will switch off. Hysteresis values are relative to setpoint.

Output (OP)

This output will be on when **Process_Val** is greater than **Setpoint** plus HystUpper, off when **Process_Val** is less than **Setpoint** minus **HystLower,** and will retain its current value when between **Setpoint** plus **HystUpper** and **Setpoint** minus **HystLower.**

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Process_Val	**DINT**	0	Oper	Oper	High Limit Low Limit	+2147483647 -2147483648
Setpoint	**DINT**	0	Oper	Oper	High Limit Low Limit	+2147483647 -2147483648
HystUpper	**DINT**	1	Oper	Oper	High Limit Low Limit	+2147483647 0
HystLower	**DINT**	1	Oper	Oper	High Limit Low Limit	+2147483647 0
Output	**BOOL**	0	Oper	Block	Senses	On (1) Off (0)

Table 8-6 Hysteresis Dint Parameter Attributes

HYSTERESIS TIME FUNCTION BLOCK

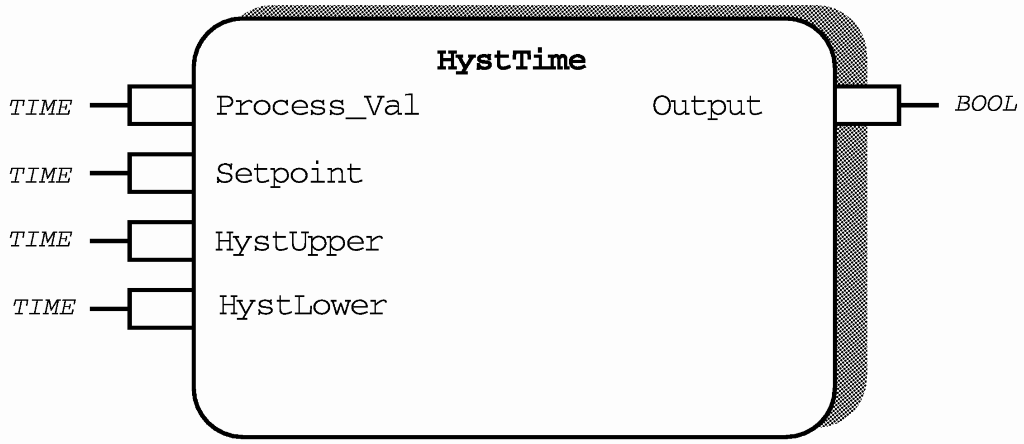


Figure 8-10 Hystersis Time Function Block Diagram

Functional Description

When monitoring analogue values there are many situations where some form of hysteresis is required. This function block allows the independent setting of an upper and lower hysteresis point on either side of a **Setpoint** for values of type Time. When the **Process_Val** goes above **Setpoint** plus the upper hysteresis limit, HystUpper, the **Output** switches on and stays on until **Process_Val** goes below Setpoint minus the lower hysteresis limit, HystLower.

By adjusting the upper and lower hysteresis values it is possible to create a number of different strategies for causing the output to activate or de-activate. By setting HystUpper to equal **Setpoint,** classic high alarm hysteresis can be created: **Output** becomes true when **Setpoint** is exceeded, and only goes false again when the **Process_Val** has fallen lower than **Setpoint** minus HystLower.

By setting HystLower to zero, and inverting the **Output** in subsequent wiring statements, classic low alarm hysteresis can be created: the **Output** becomes false when **Process_Val** goes below **Setpoint,** and only goes true again when the input has risen higher than **Setpoint** plus HystUpper. Other possibilities arise by using HystUpper and HystLower in tandem.

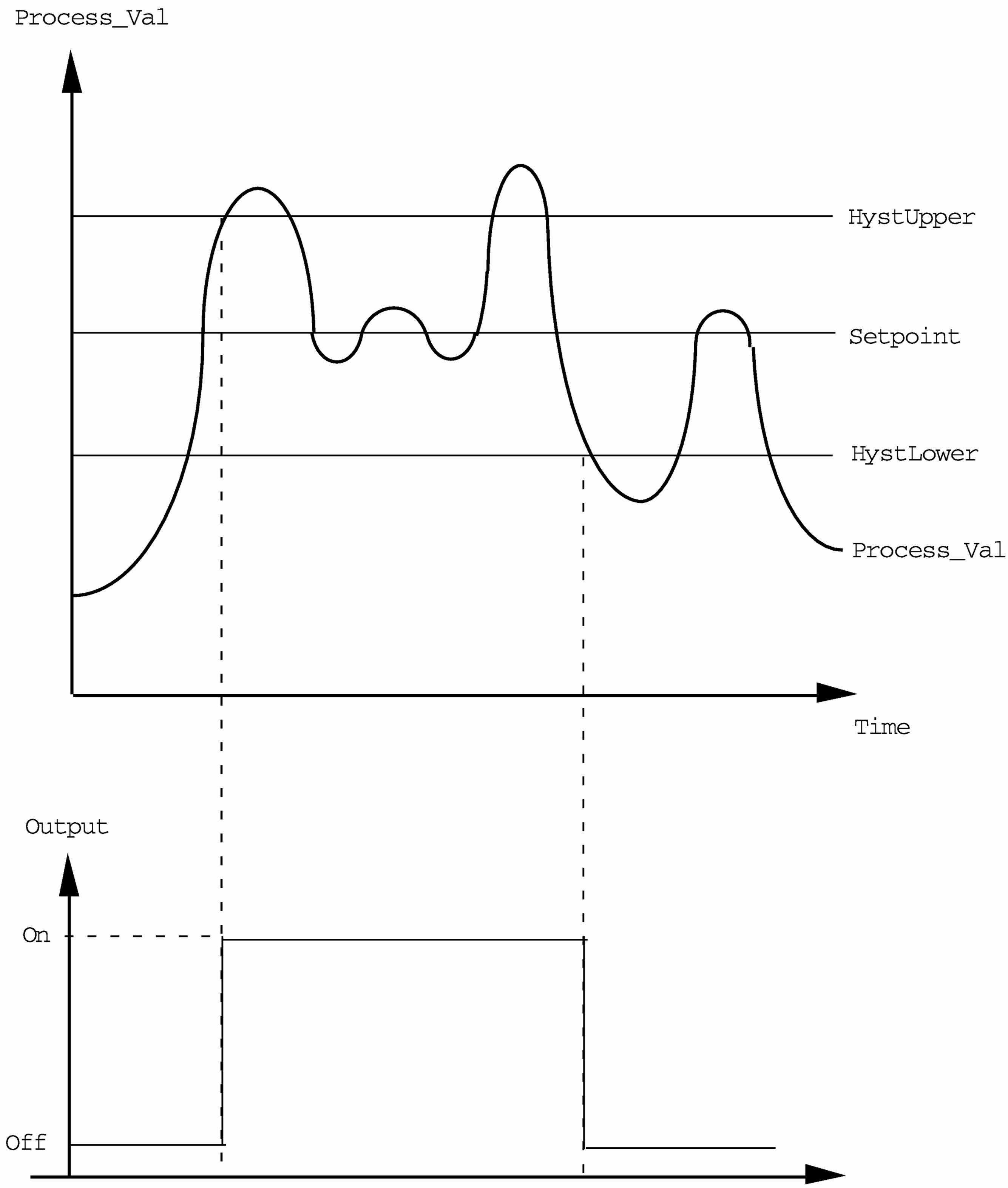


Figure 8-11 Upper and Lower Hysteresis

Function Block Attributes

Type:..1C64
Class: ....CONDITION
Default Task: ....Task_2
Short List: ....Process_Val, Setpoint, Output
Memory Requirements: ....18 Bytes

Parameter Descriptions

Process_Val (PV)

The value to be compared against the Setpoint. The variable on which an alarm is required would be wired to this input.

Setpoint (SP)

The principal value against which **Process_Val** is to be compared.

HystUpper (HYU)

The positive offset from setpoint which is the level at which Output will switch on. Hysteresis values are relative to setpoint.

HystLower (HYL)

The negative offset from setpoint which is the level at which Output will switch off. Hysteresis values are relative to setpoint.

Output (OP)

This output will be on when **Process_Val** is greater than Setpoint plus HystUpper, off when **Process_Val** is less than Setpoint minus HystLower, and will retain its current value when between Setpoint plus HystUpper and Setpoint minus HystLower.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Process_Val	**TIME**	0ms	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
Setpoint	**TIME**	0ms	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
HystUpper	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
HystLower	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
Output	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)

Table 8-7 Hysteresis Time Parameter Attributes

TOLERANCE REAL FUNCTION BLOCK

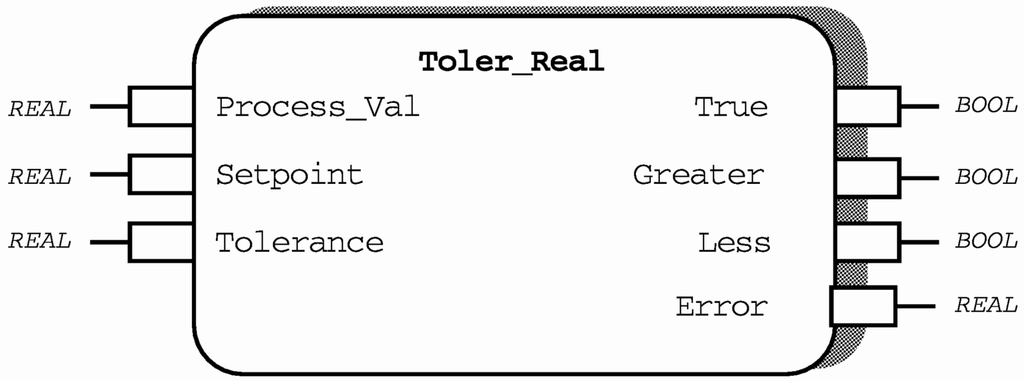


Figure 8-12 Tolerance Real Function Block Diagram

Functional Description

The difference between the incoming **Process_Val** and a **Setpoint** is presented as the Error output. The **Process_Val** is also compared to **Setpoint** plus and minus Tolerance. If the **Process_Val** is greater than **Setpoint** plus Tolerance, then only the output Greater is true. If the **Process_Val** is less than **Setpoint** minus Tolerance, then only the output Less is true. Whilst **Process_Val** is within **Setpoint** plus and minus Tolerance the output True is true, and Greater and Less are false.

Thus this function block provides in one function the information necessary for some simple control action, providing the absolute error, a dead band around setpoint and too high / too low indicators.

It is for use with variables of type Real.

Function Block Attributes

Type: 1C70

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Process_Val, Setpoint, Tolerance, True

Memory Requirements: ....20 Bytes

Parameter Descriptions

Process_Val (PV)

The value to be compared with **Setpoint** and **Tolerance.** The **Process_Val** from an input function block would normally be wired to this.

Setpoint (SP)

The central value of the tolerance band, and the value used to calculate Error in conjunction with **Process_Val.**

Tolerance (TOL)

The width of the band around **Setpoint** for which **True** is on. This corresponds to the dead band in simple increase/decrease control.

True (T)

This output is true whilst the **Process_Val** lies within **Setpoint** plus or minus **Tolerance.**

True=Process_Val<(Setpoint+Tolerance)
AND Process_Val>(Setpoint-Tolerance)

Greater (G)

When **Process_Val** is greater than or equal to **Setpoint** plus **Tolerance** this output will be true.

Greater=Process_Val>=(Setpoint+Tolerance)

Less (L)

When **Process_Val** is less than or equal to **Setpoint** minus **Tolerance** this output will be true.

Less=Process_Val<=(Setpoint-Tolerance)

Error (E)

Indicates the absolute difference between **Process_Val** and **Setpoint.**

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Process_Val	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Setpoint	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Tolerance	**REAL**	1	Oper	Oper	High Limit Low Limit	+3·402823E+38 0
True	**BOOL**	On (1)	Oper	Block	Senses	Off (0) On (1)
Grater	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)
Less	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)
Error	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 0

Table 8-8 Tolerance Real Parameter Attributes

TOLERANCE DINT FUNCTION BLOCK

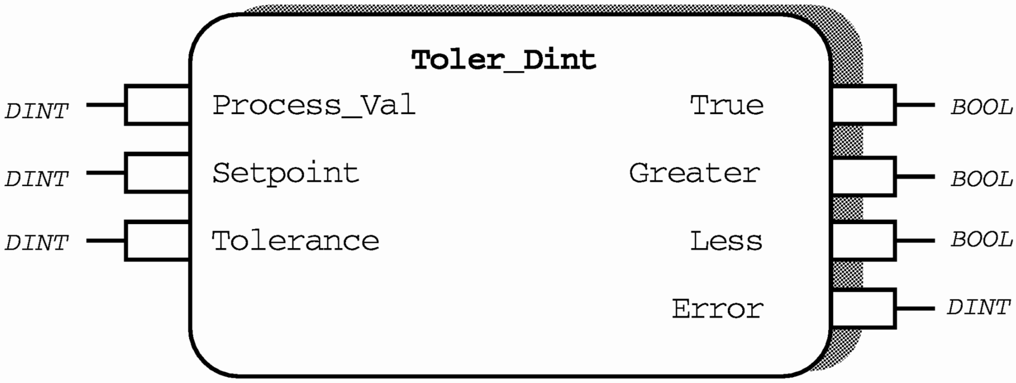


Figure 8-13 Tolerance Dint Function Block Diagram

Functional Description

The difference between the incoming **Process_Val** and a **Setpoint** is presented as the **Error** output. The **Process_Val** is also compared to **Setpoint** plus and minus **Tolerance**. If the **Process_Val** is greater than **Setpoint** plus **Tolerance,** then only the output Greater is true. If the **Process_Val** is less than **Setpoint** minus **Tolerance,** then only the output Less is true. Whilst **Process_Val** is within **Setpoint** plus and minus **Tolerance** the output True is true, and Greater and Less are false.

Thus this function block provides in one function the information necessary for some simple control action, providing the absolute error, a dead band around setpoint and too high / too low indicators.

It is for use with variables of type Dint.

Function Block Attributes

Type: 1C72

Class: .... CONDITION

Default Task: .... Task_2

Short List: .... Process_Val, Setpoint, Tolerance, True

Memory Requirements: .... 20 Bytes

Parameter Descriptions

Process_Val (PV)

The value to be compared with **Setpoint** and Tolerance. The **Process_Val** from an input function block would normally be wired to this.

Setpoint (SP)

The central value of the tolerance band, and the value used to calculate Error in conjunction with **Process_Val.**

Tolerance (TOL)

The width of the band around **Setpoint** for which True is on. This corresponds to the dead band in simple increase/decrease control.

True (T)

This output is true whilst the **Process_Val** lies within **Setpoint** plus or minus **Tolerance.**

True=Process_Val<(Setpoint+Tolerance)
AND Process_Val>(Setpoint-Tolerance)

Greater (G)

When **Process_Val** is greater than or equal to **Setpoint** plus **Tolerance** this output will be true.

Greater=Process_Val>=(Setpoint+Tolerance)

Less (L)

When **Process_Val** is less than or equal to **Setpoint** minus **Tolerance** this output will be true.

Less=Process_Val<=(Setpoint-Tolerance)

Error (E)

Indicates the absolute difference between **Process_Val** and **Setpoint.**

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Process_Val	**DINT**	0	Oper	Oper	High Limit Low Limit	+2147483647 -2147483648
Setpoint	**DINT**	0	Oper	Oper	High Limit Low Limit	+2147483647 -2147483648
Tolerance	**DINT**	1	Oper	Oper	High Limit Low Limit	+2147483647 0
True	**BOOL**	On (1)	Oper	Block	Senses	Off (0) On (1)
Grater	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)
Less	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)
Error	**DINT**	0	Oper	Block	High Limit Low Limit	+2147483647 0

Table 8-9 Tolerance Dint Parameter Attributes

TOLERANCE TIME FUNCTION BLOCK

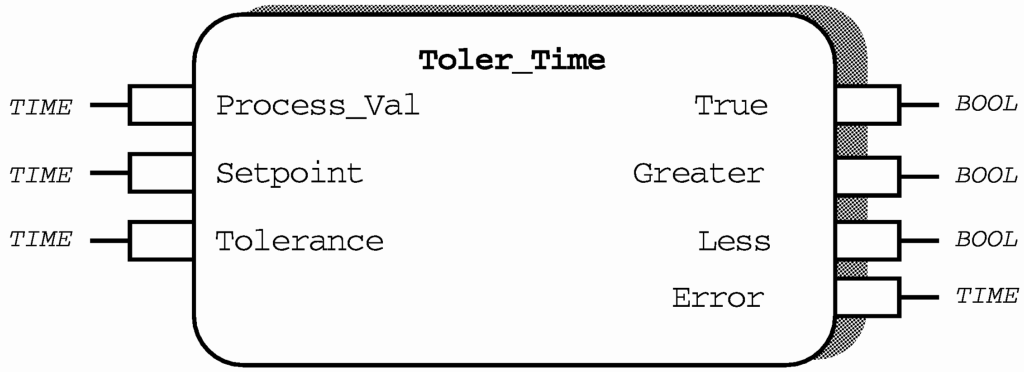


Figure 8-14 Tolerance Time Function Block Diagram

Functional Description

The difference between the incoming **Process_Val** and a **Setpoint** is presented as the Error output. The **Process_Val** is also compared to **Setpoint** plus and minus Tolerance. If the **Process_Val** is greater than **Setpoint** plus Tolerance, then only the output Greater is true. If the **Process_Val** is less than **Setpoint** minus Tolerance, then only the output Less is true. Whilst **Process_Val** is within **Setpoint** plus and minus Tolerance the output True is true, and Greater and Less are false.

Thus this function block provides in one function the information necessary for some simple control action, providing the absolute error, a dead band around **Setpoint** and too high / too low indicators.

It is for use with variables of type Time.

Function Block Attributes

Type: 1C74

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Process_Val, Setpoint, Tolerance, True

Memory Requirements: ....20 Bytes

Parameter Descriptions

Process_Val (PV)

The value to be compared with **Setpoint** and **Tolerance**. The **Process_Val** from an input function block would normally be wired to this.

Setpoint (S)

The central value of the tolerance band, and the value used to calculate Error in conjunction with **Process_Val.**

Tolerance (TOL)

The width of the band around **Setpoint** for which **True** is on. This corresponds to the dead band in simple increase/decrease control.

True (T)

This output is true whilst the **Process_Val** lies within **Setpoint** plus or minus Tolerance.

True=**Process_Val**<(Setpoint+Tolerance)

AND **Process_Val**>(Setpoint-Tolerance)

Greater (G)

When **Process_Val** is greater than or equal to **Setpoint** plus **Tolerance** this output will be true.

Greater=Process_Val>=(Setpoint+Tolerance)

Less (L)

When **Process_Val** is less than or equal to **Setpoint** minus **Tolerance** this output will be true.

Less=Process_Val<=(Setpoint-Tolerance)

Error (E)

Indicates the absolute difference between **Process_Val** and **Setpoint.**

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Process_Val	**TIME**	0ms	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
Setpoint	**TIME**	0ms	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
Tolerance	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0
True	**BOOL**	On (1)	Oper	Block	Senses	Off (0) On (1)
Grater	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)
Less	**BOOL**	Off (0)	Oper	Block	Senses	On (1) Off (0)
Error	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0

Table 8-10 Tolerance Time Parameter Attributes

RELATIVE HUMIDITY AND DEW POINT FUNCTION BLOCK

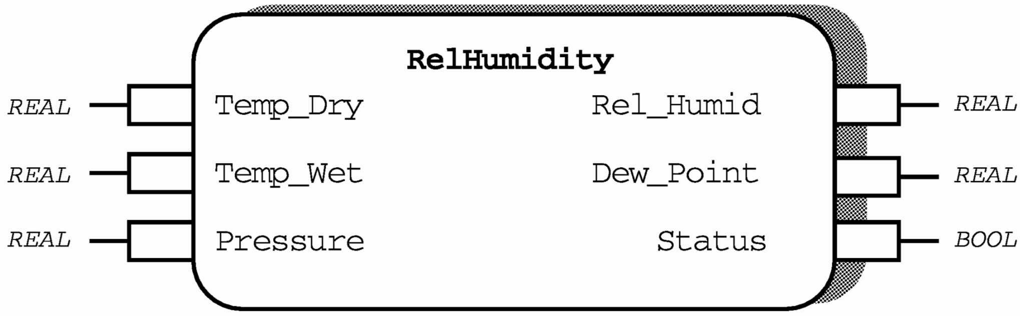


Figure 8-15 Relative Humidity and Dew Point Function Block Diagram

Functional Description

This function block provides the means of calculating the **Relative Humidity** and **Dew Point** from two temperatures using the standard technique using wet and dry thermosensor bulbs. The two temperatures would normally be brought into PC3000 using any two analogue input channels. Standard formulae are used to calculate Relative Humidity and Dew Point from the Wet and Dry bulb temperatures. For more information see BS 4833:1986, BS1399 and also the manuals for the Eurotherm Controls Ltd 900EPC and Chessel Corporation 390 Recorder. All temperatures units are fixed as degrees Celsius.

Different atmospheric pressures are catered for using the Pressure input pin (in mBar).

Function Block Attributes

Type: 1C80

Class: .... CONDITION

Default Task: .... Task_2

Short List: .... Temp_Dry, Temp_Wet,
Rel_Humid, ... Dew_Point

Memory Requirements: .... 34 Bytes

Parameter Descriptions

Temp_Dry

The temperature of the dry bulb sensor should be wired to this input. The input conditioning used should result in a value in degrees Celsius being presented to this input since the formulae used only allow for this.

Temp_Wet

The temperature of the wet bulb sensor should be wired to this input. The input conditioning used should result in a value in degrees Celsius being presented to this input since the formulae used only allow for this.

Pressure

Pressure from an atmospheric pressure sensor close to the thermo-sensors should be wired to this input. The value must be in millibar.

Rel_Humid

The relative humidity of the air surrounding the thermo-sensors is indicated by this output as a percentage. Refer to BS 4833:1986 and BS1399 for the formulae used in this calculation.

If the Wet bulb exceeds 100 Deg C (i.e. boiling) or is less than 0 Deg C (i.e.frozen), then the **Relative Humidity** will be clamped to 100% or 0% respectively. If an illegal temperature combination results in a relative humidity of greater than 100% or less than 0%, **Rel_Humid** will clamp to 100% or 0% respectively.

Dew_Point

The dewpoint temperature calculated from the wet and dry bulb temperatures and the ambient pressure. If the **Wet bulb** exceeds 100 Deg C (i.e. boiling) or is less than 0 Deg C (i.e.frozen), then the **Dew Point** will be clamped to the Wet bulb temperature or 0 Deg C respectively.

If an illegal temperature combination results in a relative humidity of less than 0% the **Dew_Point** will be forced to 0 Deg C.

Status

This output indicates the acceptability or otherwise of the input values.

Go(1): The input values are within limits and the output values of dewpoint and relative humidity are valid.

Nogo(0): An illegal temperature or temperature/pressure combination is being presented at the inputs causing calculations to fail.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Temp_Dry	**REAL**	26	Oper	Oper	High Limit Low Limit	100 0
Temp_Wet	**REAL**	25	Oper	Oper	High Limit Low Limit	100 0
Pressure	**REAL**	1013	Oper	Oper	High Limit Low Limit	1200 800
Rel_Humid	**REAL**	92	Oper	Oper	High Limit Low Limit	100 0
Dew_Point	**REAL**	25	Oper	Oper	High Limit Low Limit	100 0
Status	**BOOL**	Go(1)	Oper	Oper	Senses	NOGO()) Go(1)

Table 8-11 Relative Humidity and Dew Point Parameter Attributes

RATE_LIMIT FUNCTION BLOCK

Prior to version 3.00 this function block was located in the class OTHERS

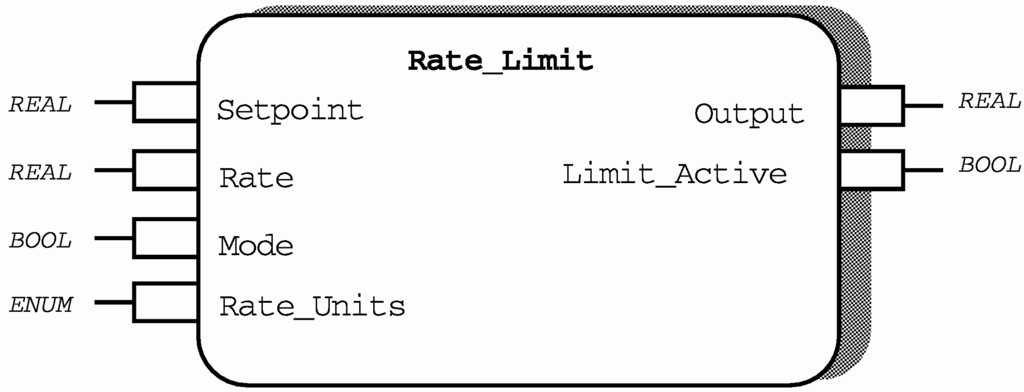


Figure 8-16 Rate_Limit Function Block

Functional Description

The **Rate_Limit** function block is used to limit the maximum rate of change of a parameter. The parameter to be rate limited is input to the **Setpoint** and the rate limited value of the parameter is output through Output. The maximum allowed rate of change of Output is defined by Rate, with the units of Rate being defined by the parameter **Rate_Units.** When rate limiting is occuring, **Limit_Active** will be set to Limit (1). The function block has two modes of operation, which are defined by the parameter Mode.

Modes of Operation

Track (0): in Track mode, the Output follows the Setpoint without any rate limiting.

Limit (1): in Limit mode, the maximum rate of change of Output is limited to the value set by Rate.

Function Block Attributes

Type:..ICA0

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Setpoint, Mode, Output

Memory Requirement:...........................32 Bytes

Execution Time: ....298 µ Secs

Parameter Descriptions

Setpoint (SP)

The **Setpoint i**s the input to the function block which is to be rate limited.

Rate (R)

The **Rate** defines the maximum rate of change to which the Output is to be limited. The units of Rate are defined by **Rate_Units.**

Mode (M)

The **Mode** defines the mode of operation of the function block.as described earlier.

Rate_Units (RU)

The **Rate_Units** defines the units for the **Rate** parameter. **Rate_Units** can be set to four possible states:

/Second (0) :	The rate will be per second
/Minute (1):	The rate will be per minute
/Hour (2):	The rate will be per hour
/Day (3):	The rate will be per day

Output (OP)

The **Output** is the rate limited output of the function block. In Track (0) mode, the **Output** will follow the **Setpoint** without rate limiting being implemented. In Limit(1) mode, the **Output** will track the **Setpoint** with its maximum rate of change being limited to the value set by **Rate.**

Limit_Active (LA)

Limit_Output is an indicator to denote that rate limiting is taking place. If the rate limiter is active **Output** will be different from Setpoint and **Limit_Output** will be set to Limit (1). If **Output** is equal to **Setpoint,** the rate limiter will not be active, so **Limit_Output** will be set to Track (0).

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Limit_Active	**BOOL**	Track (0)	Oper	Block	Senses	Track (0) Limit (1)
Mode	**BOOL**	Track (0)	Oper	Config	Senses	Track (0) Limit (1)
Output	**REAL**	0.0	Oper	Block	High Limit Low Limit	10,000 -10,000
Rate	**REAL**	0.0	Oper	Oper	High Limit Low Limit	1,000 0
Rate_Units	**ENUM**	/ Second (0)	Oper	Config	Senses	/ Second (0) / Minute (1) / Hour (2) / Day (3)
Setpoint	**REAL**	0.0	Oper	Oper	High Limit Low Limit	10,000 -10,000

Table 8-12 Rate_Limit Parameter Attributes

FREQUENCY COUNTER FUNCTION BLOCK

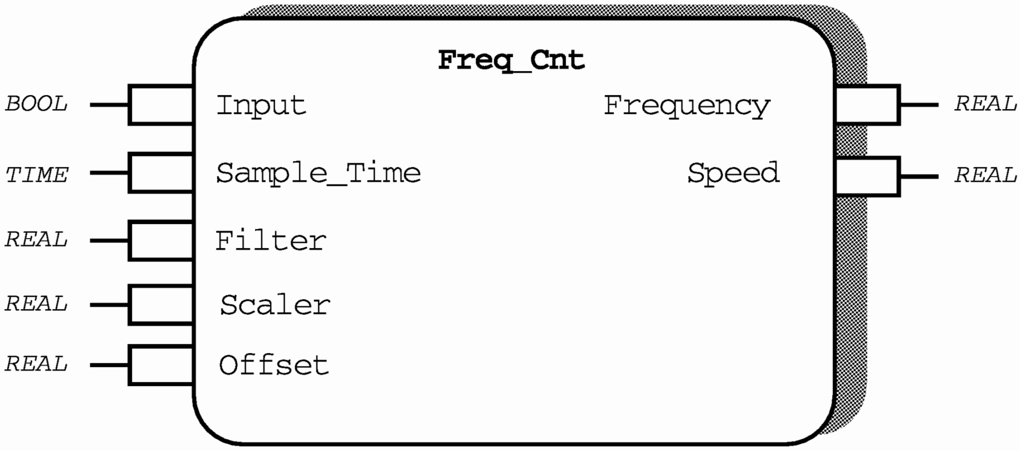


Figure 8-17 Frequency Counter Function Block

Functional Description

The **Freq_Cnt** function block accepts a single boolean input and provides an output corresponding to the frequency of the boolean input.

Multiple digital inputs such as two in quadrature can be handled in the wiring to the input of the block.

Scaling of the frequency measurement to provide an output in engineering units of speed is also provided.

It should be noted that Digital Inputs can only be scanned at 5ms maximum rate which gives a theoretical maximum frequency of 100Hz. Because of uncertainties in execution time, the actual maximum is realistically 60-70Hz. Whatever the scan rate of the digital input, the **Freq_Meter** function block should be executed at the fastest possible rate in order to provide the best possible resolution in timing measurements.

Function Block Attributes

Type: ICC8

Class:.. CONDITION

Default Task: .... Task_1

Short List: .... Input, Sample_Time, Frequency, Speed

Memory Requirement:......................... 48 Bytes

Parameter Descriptions

Input (IN)

This is the pulse input whose frequency is to be measured.

Sample_Time (ST)

The **Sample_Time** input specifies the update rate of the fequency output and the speed of response to changing frequencies. The **Sample_Time** needs to be chosen carefully to compromise between update speed of the output, for control and ergonomic reasons, and smoothness and accuracy of the result. It MUST be long enough to guarantee at least one pulse in the **Sample_Time**, at the slowest measurable speed. If no pulses are received in the Sample_Time the block sets the Frequency output to zero.

Filter (FLT)

The **Filter** input specifies the filter coefficient which will be used in the frequency output filter. To a first approximation, the time constant of the output filter is given by the expression

Time Constant = Sample_Time * Filter

Scaler (SCL) and Offset (OFS)

These parameters are used by the function block to calculate the **Speed** output value. The **Speed** output is given by the expression

Speed = Frequency * Scaler + Offset

Frequency (FRQ)

The **Frequency** output indicates the number of pulses /second which are being applied to the **Input.** The units of this parameter are fixed to Hz.

Speed (SPD)

This parameter relates frequency to engineering units of speed. The **Scaler** and **Offset** parameters are used to define the coefficients of the conversion. **Speed** is related to frequency by the expression

Speed = Frequency * Scaler + Offset

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Input	**BOOL**	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Sample_Time	**TIME**	2s	Oper	Oper	High Limit Low Limit	23d23h59m59s999m s 0s
Filter	**REAL**	1.0	Oper	Oper	High Limit Low Limit	+3·402823E+38 0
Scaler	**REAL**	1.0	Oper	Oper	High Limit Low Limit	+3·402823E+38 0
Offset	**REAL**	0.0	Oper	Oper	High Limit Low Limit	+3402823E+38 -3·402823E+38
Frequency	**REAL**	0.0	Oper	Block	High Limit Low Limit	+3·402823E+38 0
Speed	**REAL**	0.0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38

Table 8-13 Frequency Counter Parameter Attributes

ASTABLE CYCLE TIME FUNCTION BLOCK

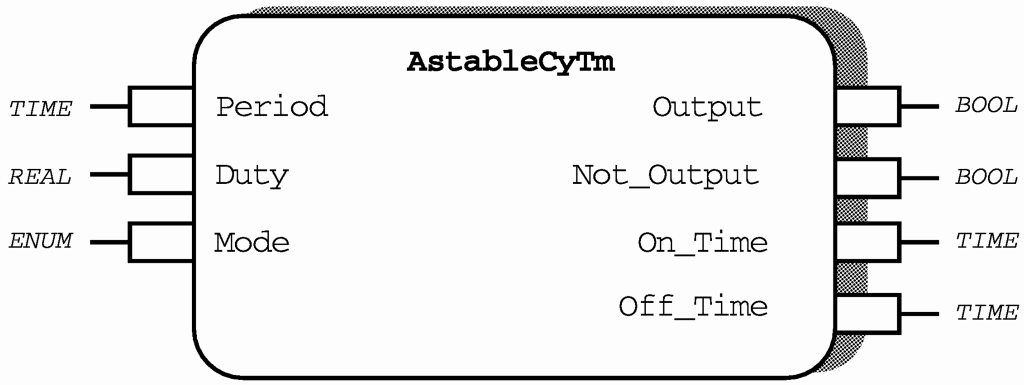


Table 8-18 Astable Cycle Time Function Block

Functional Description

A regular pulse train is produced on the output by specifying the repeat time (Period) and the percentage of this **Period** that the function block **Output** should be switched on (Duty). The **Period** and the Duty may be varied to give pulses of the desired length which recur at a regular interval.

By way of example, suppose a **Period** of 4s is selected and then Duty is set to 50% - the **Output** will be on for 2s and then off for 2s. This pattern will then continuously repeat. If the Duty is changed to 25% the **Output** will be on for 1s and then off for 3s.

Function Block Attributes

Type: 1CCA

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Period, Duty, Mode, Output

Memory Requirements: ....32 Bytes

Parameter Descriptions

Period (PER)

This is the total time for the on/off cycle. It may also be considered as the repeat interval of the function block.

Duty (DTY)

The percentage of the **Period** for which the **Output** is switched on, and the **Not_Output** is switched off.

Mode (M)

The Mode parameter is used to control the operation of the function blocks:

Reset: all internal timers set back to zero, **Output** set Off(0) and **Not_Output** set On(1).

Run: **Output** and **Not_Output** update in accordance with settings of **Period** and **Duty.**

Hold: **Output** and **Not_Output** hold present values and internal timers are held. Returning the **Mode** to **Run** will cause the outputs to continue to update from where they left off.

Wait: the current part of the cycle is completed, so **Output** and **Not_Output** change state once more, and then **Output** and **Not_Output** hold new values and internal timers are held. Returning the **Mode** to Run will cause the outputs to continue to update from the last change of state.

Output (OP)

When the function block is switched into Run mode, the output remains switched off for a time equal to Period*(100-Duty), and then on for a time equal to Period*Duty. This sequence then repeats.

Not_Output (NOP)

The **Not_Output** is always the inverse condition of the **Output.**

On_Time (ONT)

This specifies the length of time for which the output will be switched on during this on-period. When the **Duty** or **Period** are changed, the value updates when the next on-period commences.

Off_Time (OFT)

This specifies the length of time for which the output will be switched off during this off-period. When the **Duty** or **Period** are changed, the value updates when the next off-period commences.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Period	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0ms
Duty	**REAL**	50%	Oper	Oper	High Limit Low Limit	99% 1%
Mode	**ENUM**	Reset (0)	Oper	Oper	See parameter List	
Output	**BOOL**	Off (0)	Oper	Block	Senses	Off (0) On (1)
Not_Output	**BOOL**	On (1)	Oper	Block	Senses	On (1) Off (0)
On_Time	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms
Off_Time	**TIME**	0ms	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms

Table 8-14 Astable Cycle Time Parameter Attributes

ASTABLE ON/OFF TIME FUNCTION BLOCK

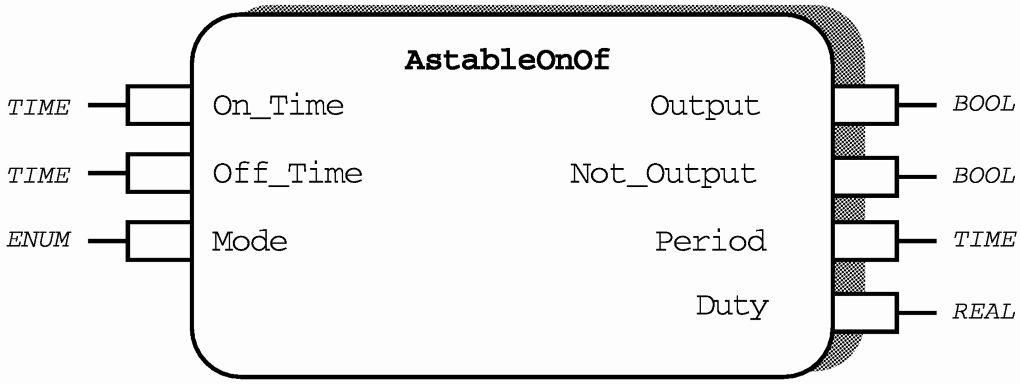


Figure 8-19 Astable On/Off Time Function Block

Functional Description

A regular pulse train is produced on the output by specifying the on time and the off time for each pulse. The pulses recur regularly. The **Period** (the sum of **On_Time** and **Off_Time**) and the duty (**On_Time** as a percentage of the period) appear as outputs.

By way of example, suppose an **On_Time** of 4s is selected and an **Off_Time** of 2s is set. The output will be on for 4s and then off for 2s. This pattern will then continuously repeat. The **Duty** will be shown as 33% and the **Period** as 6s.

Function Block Attributes

Type: 1CCC

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....On_Time, Off_Time, Mode, Output

Memory Requirements: ....32 Bytes

Parameter Descriptions

On_Time (ONT)

This specifies the length of time for which the output will be switched on during each cycle.

Off_Time (OFT)

This specifies the length of time for which the output will be switched off during each cycle.

Mode (M)

The **Mode** parameters is used to control the operation of the function block:

Reset: all internal timers set back to zero, **Output** set Off(0) and **Not_Output** set On(1).

Run: **Output** and **Not_Output** update in accordance with settings of **On_Time** and **Off_Time.**

Hold: **Output** and **Not_Output** hold present values and internal timers are held. Returning the Mode to Run will cause the outputs to continue to update from where they left off.

Wait: the current part of the cycle is completed, so **Output** and **Not_Output** change state once more, and then **Output** and **Not_Output** hold new values and internal timers are held. Returning the **Mode** to Run will cause the outputs to continue to update from the last change of state.

Output (OP)

When the function block is switched into Run mode, the output remains switched off for a time equal to **Off_Time** and then on for a time equal to **On_Time**. This sequence then repeats.

Not_Output (NOP)

The **Not_Output** is always the inverse condition of the **Output.**

Period (PER)

This is the total time for the on/off cycle. It may also be considered as the repeat interval of the function block.

Duty (DTY)

The percentage of the **Period** for which the **Output** is switched on, and the **Not_Output** is switched off.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
On_Time	**TIME**	500ms	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0ms
Off_Time	**TIME**	500ms	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0ms
Mode	**Enum**	Reset (0)	Oper	Oper	See parameter List	
Output	**BOOL**	Off (0)	Oper	Block	Senses	Off (0) On (1)
Not_Output	**BOOL**	On (1)	Oper	Block	Senses	Off(0) On (1)
Period	**TIME**	1s	Oper	Block	High Limit Low Limit	23d23h59m59s999ms 0ms
Duty	**REAL**	50%	Oper	Block	High Limit Low Limit	100% 0%

Table 8-15 Astable On/Off Time Parameter Attributes

TOGGLE FUNCTION BLOCK

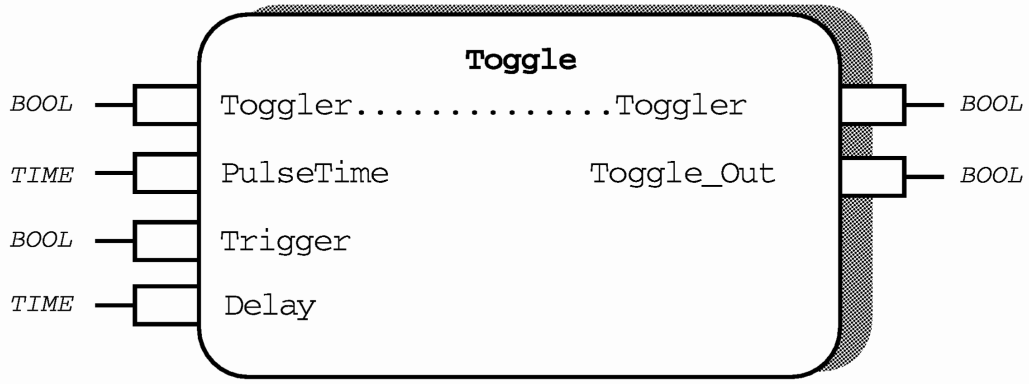


Figure 8-20 Toggler Function Block

Functional Description

The input/output parameter **Toggler** may be directly switched on and will remain on for a period specified by the **PulseTime** input. It will also come on after a delay specified by **Delay** if the input **Trigger** is set to on. In this case **Toggler** will again switch off after it has been on for a period specified by the **PulseTime** input. **Trigger** will cause **Toggler** to come on again only if **Trigger** returns to off and then back to on; in other words, the action occurs on the leading edge of **Trigger.**

Function Block Attributes

Type: 1CCE

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Toggler, Trigger, Toggle_Out

Memory Requirements: ....26 Bytes

Parameter Descriptions

Toggler (TGR)

This is an input/output. Once set to on, it will remain on for a period specified by **PulseTime**. It may be switched on directly or by means of the trigger input. In the latter case, **Toggler** switches on after **Trigger** becomes true, the delay being specified by **Delay.**

PulseTime (PT)

The period for which **Toggler** remains switched on, independent of external action.

Trigger (TRG)

Causes **Toggler** to become true after a delay specified by Delay. **Trigger** must be returned to false by the user program, and will only affect Toggler again if **Trigger** returns to false and then back to true. If **Toggler** is already on when Trigger is set to true, **Trigger** has no effect.

Delay (DLY)

The delay between **Trigger** becoming true and **Toggler** becoming true.

Toggle_Out (TGO)

This output changes State every time **Trigger** changes State from Off to On. When **Toggler** changes State from On to Off the **Toggle_Out** output does not change.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Toggler	**BOOL**	Off (0)	Oper	Oper	Senses	Off (0) On (1)
PulseTime	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0ms
Trigger	**BOOL**	Off (0)	Oper	Oper	Senses	Off (0) On (1)
Delay	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d23h59m59s999ms 0s
Toggle_Out	**BOOL**	Off (0)	Oper	Oper	Senses	Off (0) On(1)

Table 8-16 Toggler Parameter Attributes

TOTALIZER FUNCTION BLOCK

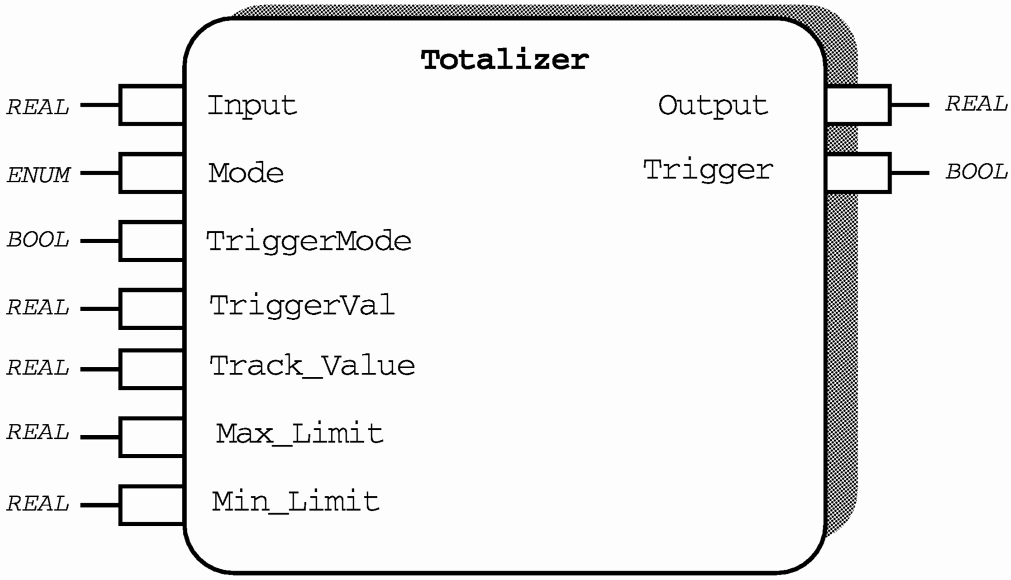


Figure 8-21 Totalizer Function Block

Functional Description

This block provides a means of totalizing an input with respect to time. It is fully bi-directional, in other words the output may increase or decrease depending on the sign of the input. The operation of the block is such that with a steady input, a value equal to **Input** is added to **Output** every one second. The task interval of the task to which the function block is assigned does not affect this. The output value is constrained within an upper and lower limit.

A **Mode** control pin allows the block to be reset and then run. **Output** may also be frozen at its current value or made to track another independent input.

A latching boolean output is provided to give an indication of either a high limit or a low limit being exceeded by the **Output.**

Function Block Attributes

Type: 1CD0
Class: ....CONDITION
Default Task: ....Task_2
Short List: ....Input, Mode, Output, Trigger.
Memory Requirements: ....46 Bytes

Parameter Descriptions

Input (IN)

The value to be totalized.

Mode (M)

Controls the operation of the function block.

Reset: **Output** is reset to zero and **Trigger i**s reset to Off.

Run: Every time the function block executes, the average of the value of the input last time the block executed and the current value of the input is added to the **Output** (trapezoidal integration). On returning to **Run** mode from Hold or Track, the **Output** begins updating from its current value.

Hold: **Output** is frozen at its current value, and none of the inputs has any effect on the function block outputs.

Track: Output follows the value of the input **Track_Value.** Changes to the inputs have no effect.

TriggerMode (TM)

Determines whether the output **Trigger** goes true when **Output** becomes greater (Upper) or less (Lower) than the **TriggerVal.**

TriggerVal (TVL)

The level at which **Trigger** becomes true, either as **Output** increases above it or decreases below it (depending on **TriggerMode).**

Track_Value (TRV)

The output value will track this input when **Mode** is set to Track**.**

Max_Limit (MAX)

The **Output** will not go above this value, either due to integrating the **Input** or through tracking the **Track_Value**

Min_Limit (MIN)

The **Output** will not go below this value, either due to integrating the **Input** or through tracking the **Track_Value**

Output (OP)

When **Mode** is Reset, **Output** will be zero.

When in Run Mode, every execution cycle the **Output** will have added to it:

$$\frac{(I(last) + I(current))/2}{t(task)}$$

where: I(last) = value of Input last time function block evaluated.

I(current) = current value of Input.

t(task) = task interval for the task to which the function block is assigned.

The only exception to this is when such addition would take **Output** beyond the limits imposed by **Max_Limit** or **Min_Limit.** In these circumstances, Output clamps at either **Max_Limit** or **Min_Limit** depending on which limit would be violated.

When in Hold Mode, **Output** will freeze at its current value.

When in Track Mode, **Output** will display whatever value is showing on the **Track_Value** input provided it is within **Max_Limit** and **Min_Limit.**

Trigger (TRG)

Trigger is a latching boolean output which is used to indicate an alarm condition occuring when in Run Mode. If **TriggerMode** is Upper, then **Trigger** becomes true when **TriggerVal** is exceeded, and then remains true until and unless Mode becomes Reset or Track when Trigger will reset to Off.

If **TriggerMode** is Lower, then **Trigger** becomes true when **Output** goes below **TriggerVal,** and then remains true until and unless **Mode** becomes Reset or Track when **Trigger** will reset to Off.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Input	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Mode	**ENUM**	Reset (0)	Oper	Oper	See Paramster List	
TriggerMode	**BOOL**	Upper (0)	Oper	Oper	Senses	Upper (0) Lower (1)
TriggerVal	**REAL**	0	Oper	Oper	High Limit Low Limit	Max_Limit Min_Limit
Track_Value	**REAL**	0	Oper	Oper	High Limit Low Limit	Max_Limit Min_Limit
Max_Limit	**REAL**	100	Oper	Oper	High Limit Low Limit	+3·402823E+38 Min_Limit
Min_Limit	**REAL**	-100	Oper	Oper	High Limit Low Limit	Max_Limit -3·402823E+38
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Trigger	**BOOL**	Off (0)	Oper	Block	Senses	Off (0) On (1)

Table 8-17 Totalizer Parameter Attributes

CUSTOM LINEARISATION FUNCTION BLOCK

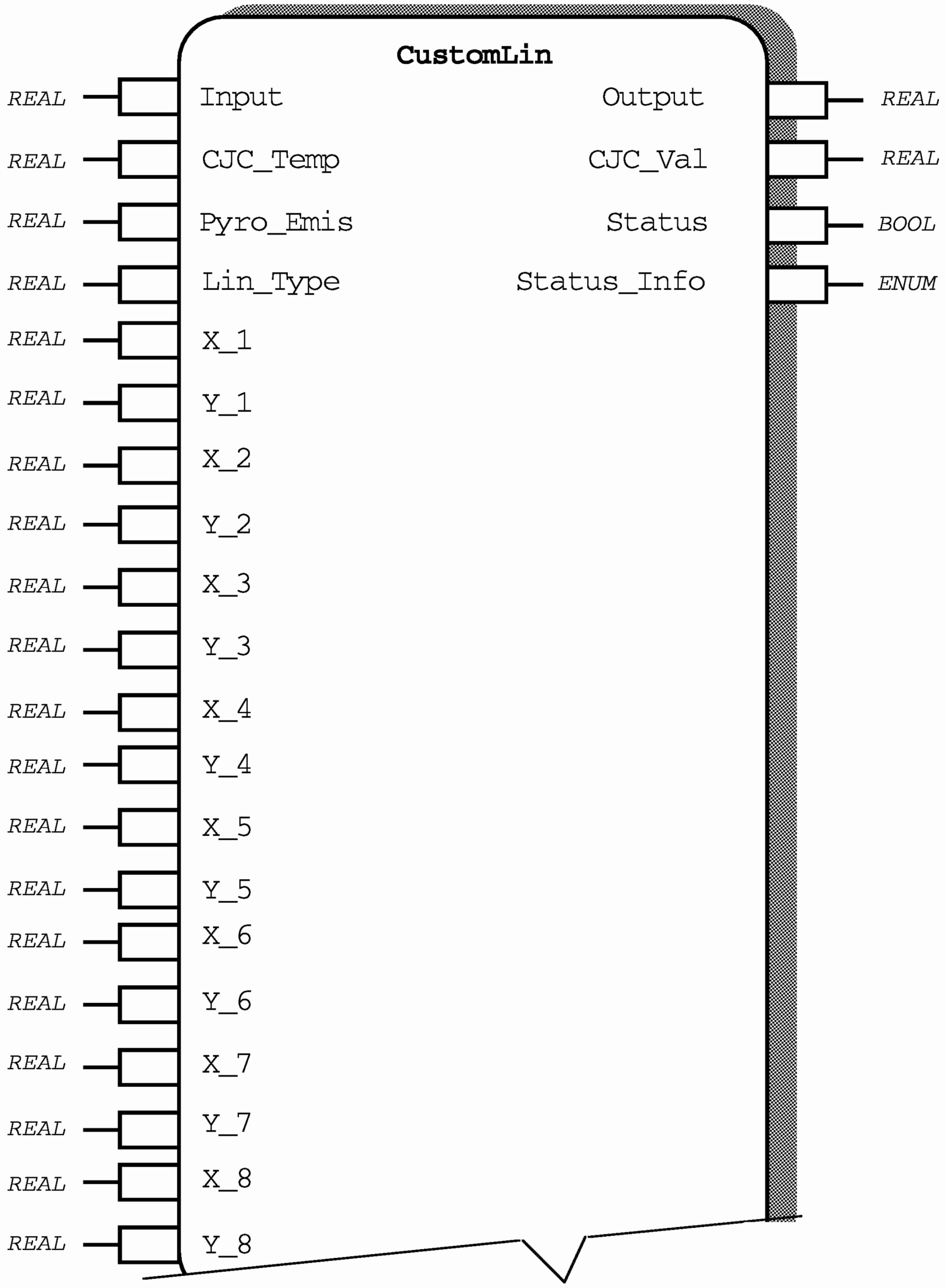


Figure 8-22 Custom Linearisation Function Block

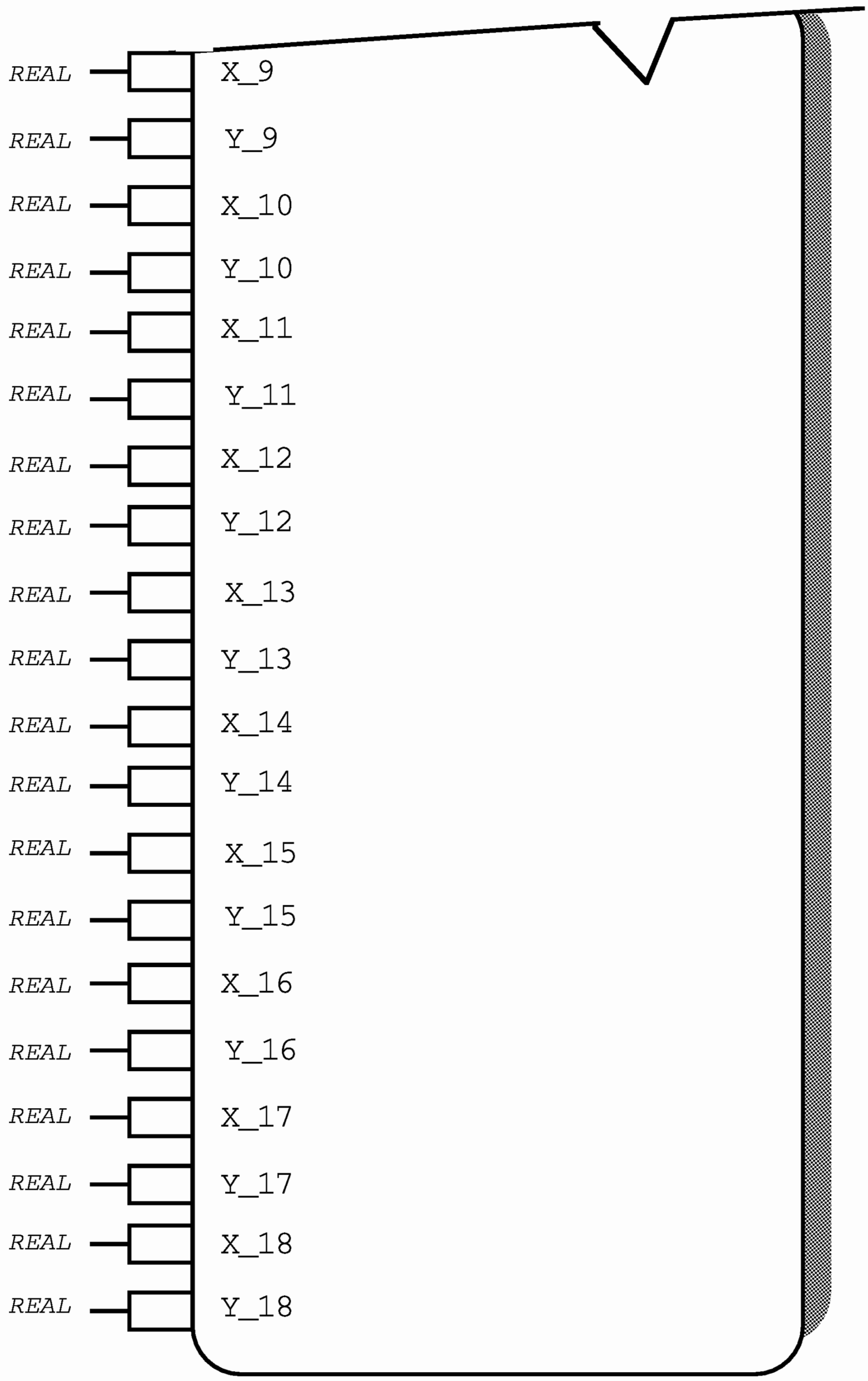


Figure 8-22 Custom Linearisation Function Block (Continued)

Functional Description

The outputs of plant sensors are often non-linear with respect to the physical variable being measured. The linearisation characteristics of the most common transducers are provided as standard with the **Analog_In** function blocks. The custom linearisation function block enables up to eighteen pairs of transducer output and the value of the physical variable to be used to define a linearisation characteristic for less common sensors. These data pairs may be obtained by calibration or from manufacturer's data.

A straight line fit is used between pairs of readings.

The values of transducer output (entered as the X values) and the related value of the physical variable (entered as the Y values) form eighteen pairs of inputs to this function block. X values are entered in order of increasing magnitude. If less than eighteen data pairs are available the table can be terminated by entering an X value that is less than its predecessor. For example, if only six data pairs are available with the largest X value being 300, then function block input X_6 would be 300 and X_7 would need to be any value less than X_6; the default of zero would do fine. However, if the six X values are all negative with the highest value (i.e. the least negative) being-90, then X_6 would be -90 and X_7 would have to be less than -90, say -91. If X_7 is left as its default of zero and Y_7 is also zero, the table will use 0,0 as a valid data pair.

When temperature measurements are being carried out, cold junction compensation (CJC) or pyrometer emmissivity can be used as part of the linearisation process. CJC is applied by entering the cold juction temperature. This value is then worked backwards through the linearisation process to give the input offset that is then applied to the Input before linearising. Emmissivity is used to scale the input value before linearising.

Function Block Attributes

Type: 1CDC

Class: ....CONDITION

Default Task: ....Task_2

Short List: ....Input, Output, Status, Status_Info.

Memory Requirements: ....198 Bytes

Parameter Descriptions

Input (IN)

The value of the plant sensor output

CJC_Temp (CJC)

The temperature of the cold junction when using a thermocouple as a sensor. This value is only used if **Lin_Type** is set to CJC.

Pyro_Emis (PEM)

The emmissivity of the body being looked at by the pyrometer. This value is only used if **Lin_Type** is set to **PyrEms.**

Lin_Type (LT)

Determines whether cold junction compensation or pyrometer emmissivity corrections are to be made.

No_CJC:	Neither CJC nor pyrometer emmissivity corrections are to be made.
CJC:	**Input** is to be corrected in line with the cold junction temperature specified by **CJC_Temp**.
PyrEms:	The input is to be scaled by the factor specified by **Pyro_Emis.**

X_1 (X1)

The first value of sensor output. The starting point of the table of sensor output vs value of physical variable measured.

Y_1 (Y1)

The first value of physical variable giving rise to sensor output X_1. The starting point of the table of sensor output vs value of physical variable measured.

X_2 (X2)

The second value of sensor output. If this value is not greater than the previous value the table will terminate on the previous value.

Y_2 (Y2)

The second value of physical variable giving rise to sensor output X_2.

Repeating for each (X,Y) pair until:

X_18 (X18)

The eighteenth value of sensor output. If this value is not greater than the previous value the table will terminate on the previous value.

Y_18 (Y18)

The eighteenth value of physical variable giving rise to sensor output X_18.

Output (OP)

The linearised value.

CJC_Val (CJV)

The offset resulting from CJC compensation.

Status (ST)

Normally this will indicate a Go status. However if **Input** is outside the bounds defined by the table (after CJC or Emmissivity correction if applicable) or if the **CJC_Temp** is outside the bounds defined by the table then Status will be NoGo.

Status_Info (STI)

Explains any status problems.

Ok:	The status is Go.
Over_R:	**Input** is above the upper bound defined by the table.
Under_R:	**Input** is below the lower bound defined by the table.
CJCOver:	**CJC_Temp** is above the upper bound defined by the table.
CJCUndr:	**CJC_Temp** is below the lower bound defined by the table.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Input	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
CJC_Temp	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Pyro_Emis	**REAL**	0	Oper	Oper	High Limit Low Limit	1 0
Lin_Type	**ENUM**	No_CJC (0)	Oper	Block	See parameter List	
X_1	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_1	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_2	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_2	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_3	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_3	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_4	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_4	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_5	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_5	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_6	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_6	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_7	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_7	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38

Table 8-16 Custom Linearisation Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
X_8	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_8	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_9	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_9	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_10	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_10	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_11	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_11	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_12	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_12	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_13	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_13	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_14	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_14	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_15	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_15	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_16	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_16	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38

Table 8-16 Custom Linearisation Parameter Attributes (continued)

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
X_17	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_17	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
X_18	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Y_18	**REAL**	0	Oper	Oper	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
CJC_Val	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38
Status	**BOOL**	NOGO (0)	Oper	Block	Senses	NOGO (0) Go (1)
Status_Info	**ENUM**	Over_R (1)	Block	Oper	See Parameter List	

Table 8-16 Custom Linearisation Parameter Attributes (continued)

RANDOM FUNCTION BLOCK

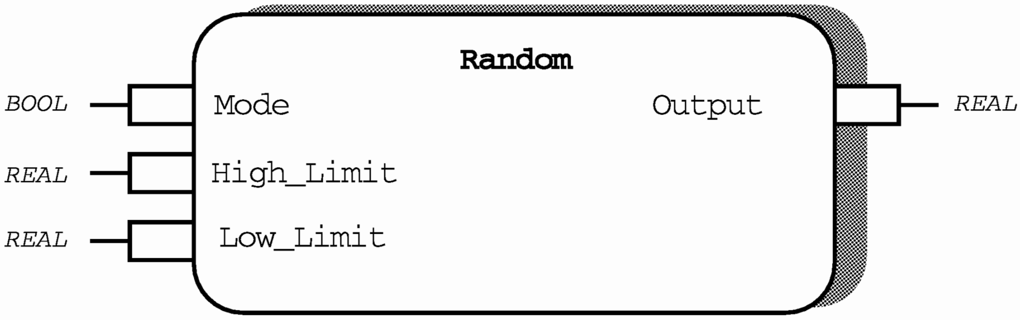


Figure 8-18 Random Function Block Diagram

Functional Description

This function block continuously generates random numbers in a specified range. An input is available to reset the random number algorithm seed.

Function Block Attributes

Type: 1CFA

Class:.. CONDITION

Default Task: .... Task_2

Short List: .... Low_Limit, High_Limit, Mode, Output.

Memory Requirements: .... 26 Bytes

Parameter Descriptions

Mode (M)

In Run mode, the function block generates a new random number every function block execution cycle. In Reset mode, the random number algorithm seed is reset.

High_Limit (HL)

Random numbers generated will be less than or equal to the value of this input.

Low_Limit (LL)

Random numbers generated will be greater than or equal to the value of this input.

Output (OP)

The random number generated is presented on this output whilst mode is set to Run. When Mode is Reset, the **Output** is equal to the **Low_Limit**.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Mode	**BOOL**	Run (0)	Oper	Oper	Senses	Run (0) Reset (1)
High_Limit	**REAL**	1	Oper	Oper	High Limit Low Limit	+3·402823E+38 Low_Limit
Low_Limit	**REAL**	0	Oper	Oper	High Limit Low Limit	High_Limit -3·402823E+38
Output	**REAL**	0	Oper	Block	High Limit Low Limit	+3·402823E+38 -3·402823E+38

Table 8-19 Random Parameter Attributes