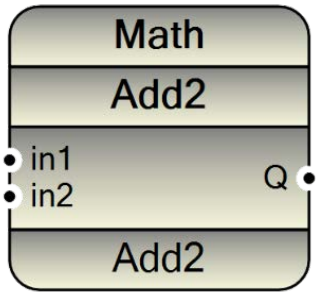
21.1 Math Group:

Mathematics Function Groups

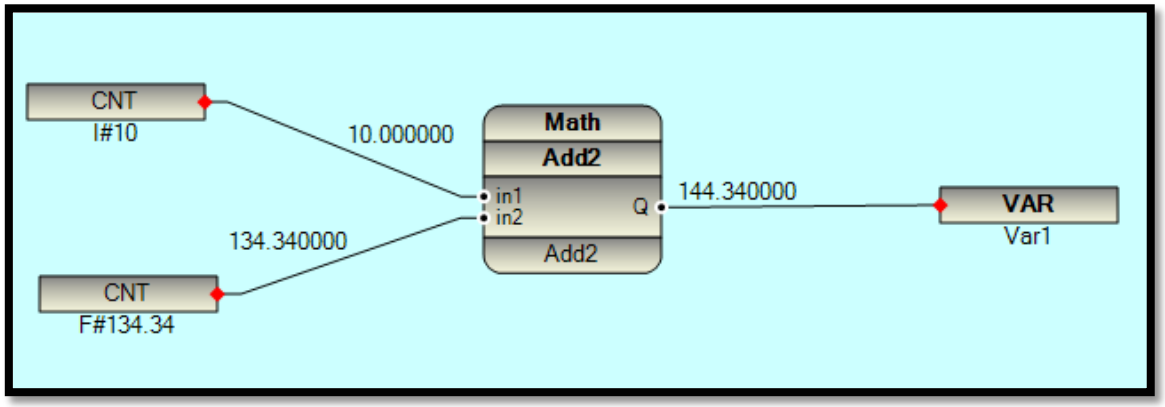
21.1.1 Add Function Block

The Add Function Block sum the values of two inputs and generated an output value. The inputs and output numerical type is ***double****.*



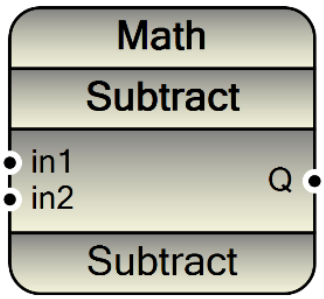
ADD		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
In2	**Double**	*B*
Q	**Double**	*A+B*

For Example:



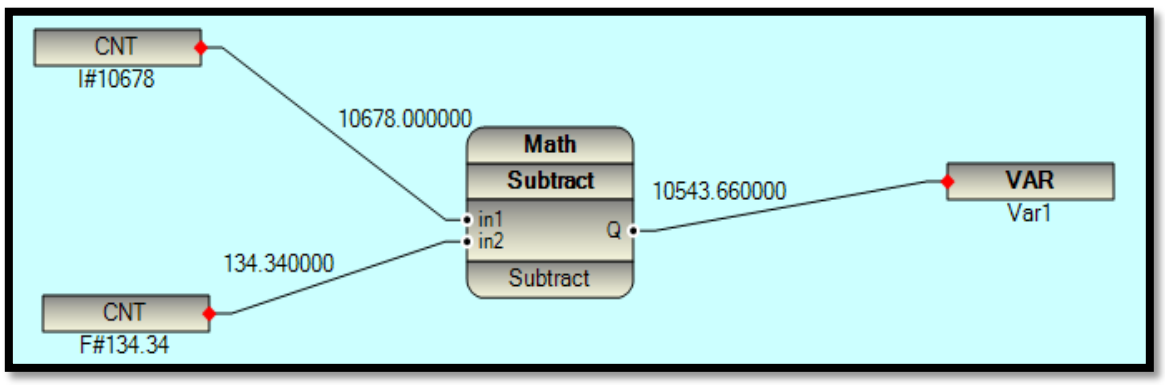
21.1.2 Subtract Function Block

The Subtract Function Block accept two inputs (in1 and in2), takes the difference, and generates an output. The inputs and output numerical type is ***double.***



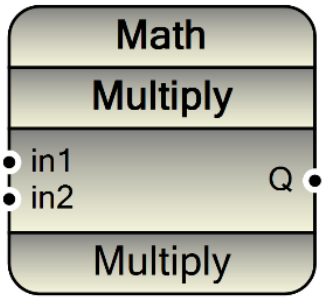
SUB		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
In2	**Double**	*B*
Q	**Double**	*A-B*

For Example:



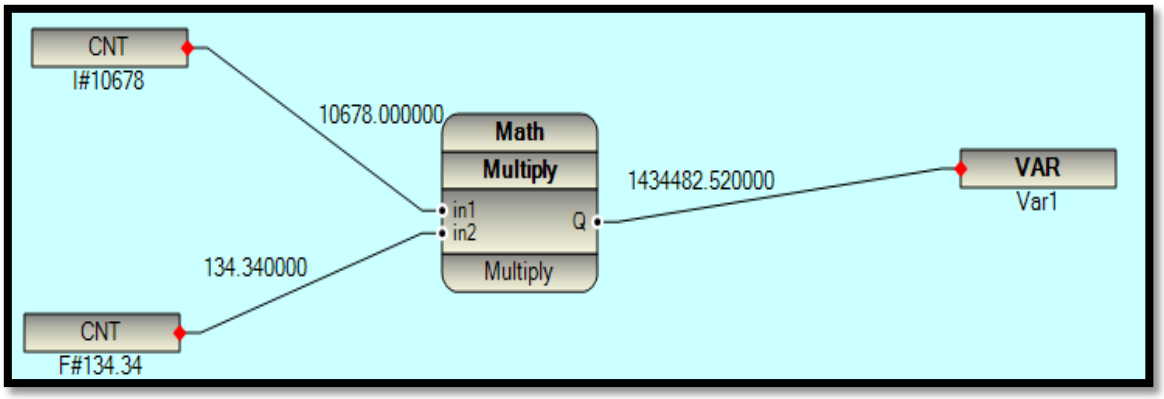
21.1.3 Multiply Function Block

The Multiply Function Block multiplies all input value connected to the block and place the result at the output. The inputs and output numerical type is ***double****.*



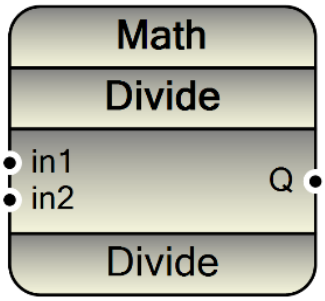
MUL		
I/O	**Data Type**	**Description**
in1	**Double**	A
In2	**Double**	B
Q	**Double**	$A \times B$

For Example:



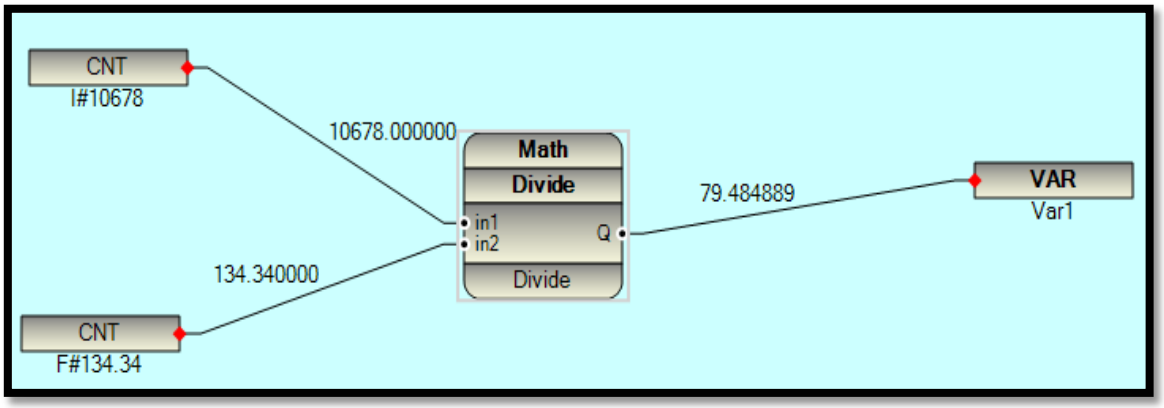
21.1.4 Divide Function Block

The Divide Function Block calculates the output as OUT=in1/in2. The inputs and output numerical type is ***double****.*



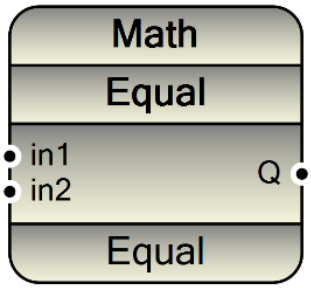
DIV		
I/O	**Data Type**	**Description**
in1	**Double**	A
In2	**Double**	B
Q	**Double**	A/B

For Example:



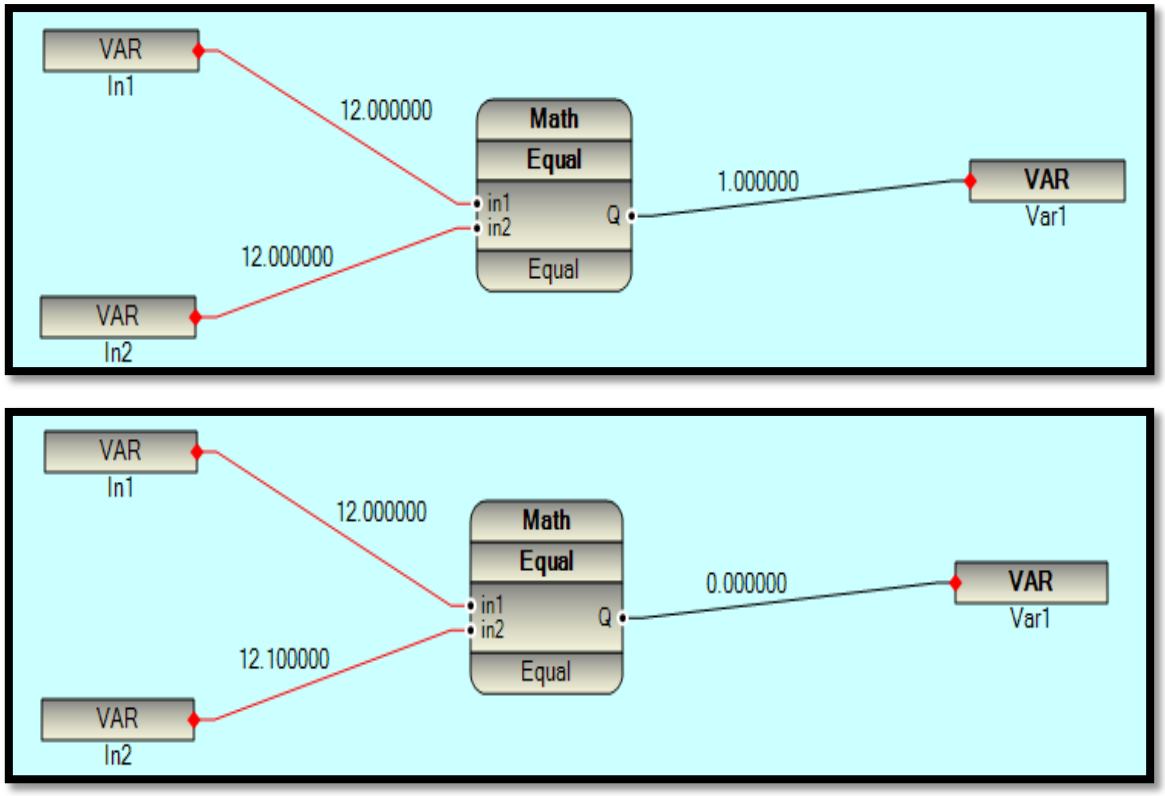
21.1.5 Equal Function Block

The Equal Function Block accept two inputs (IN1 and IN2), if the inputs value is equal set output is 1, otherwise set output is 0. The inputs and output numerical type is ***double****.*



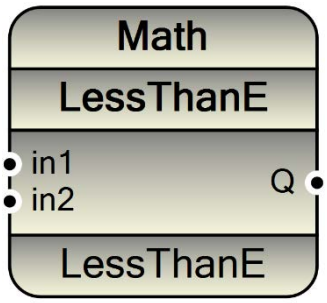
Equal		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
In2	**Double**	*B*
Q	**Double**	*If A = B → Q=1 If A<>B → Q=0*

For Example:



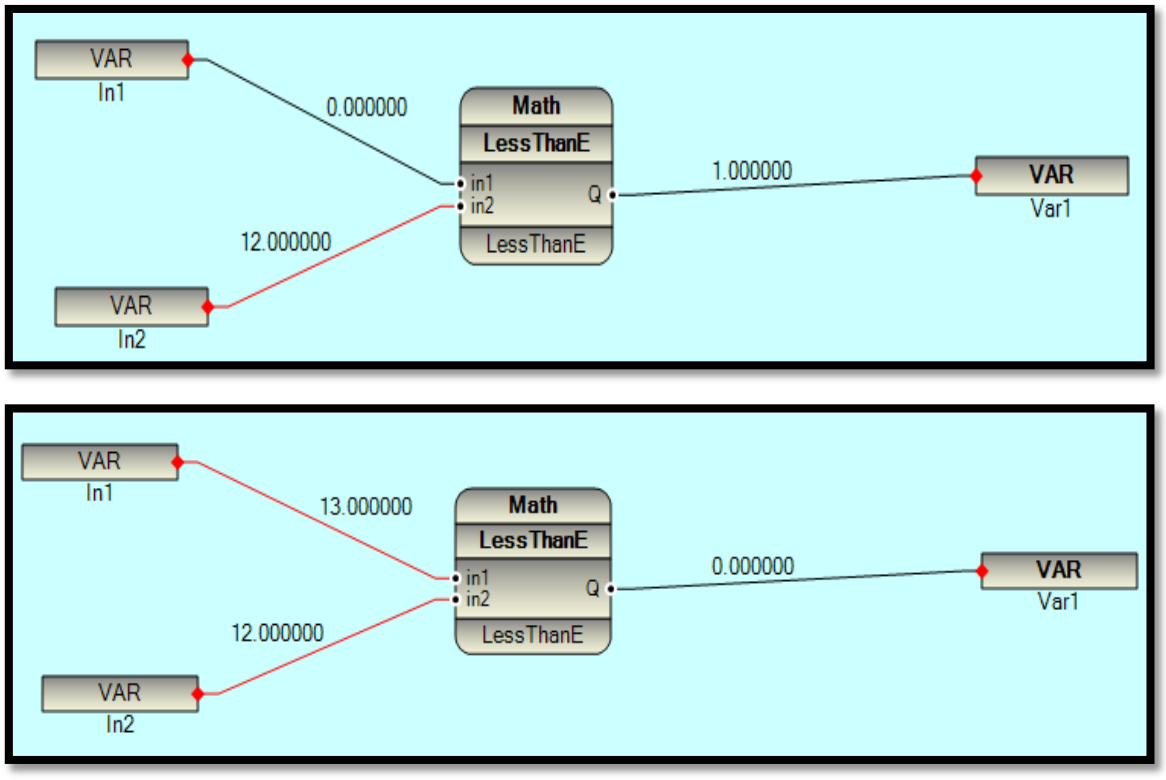
21.1.6 LessThanE Function Block

LessThanE Function Block accept two inputs (in1 and in2). If In1 is less Than or Equal to In2, Output is 1, otherwise Output is 0. The inputs and output numerical type is ***double****.*



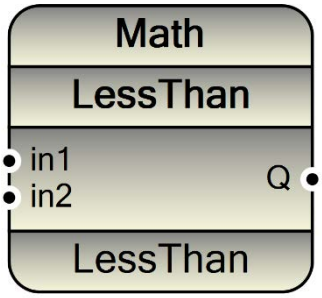
LessThanE		
I/O	**Data Type**	**Description**
in1	**Double**	A
In2	**Double**	B
Q	**Double**	$If\ A \leq B \rightarrow Q=1\ If\ A > B \rightarrow Q=0$

For Example:



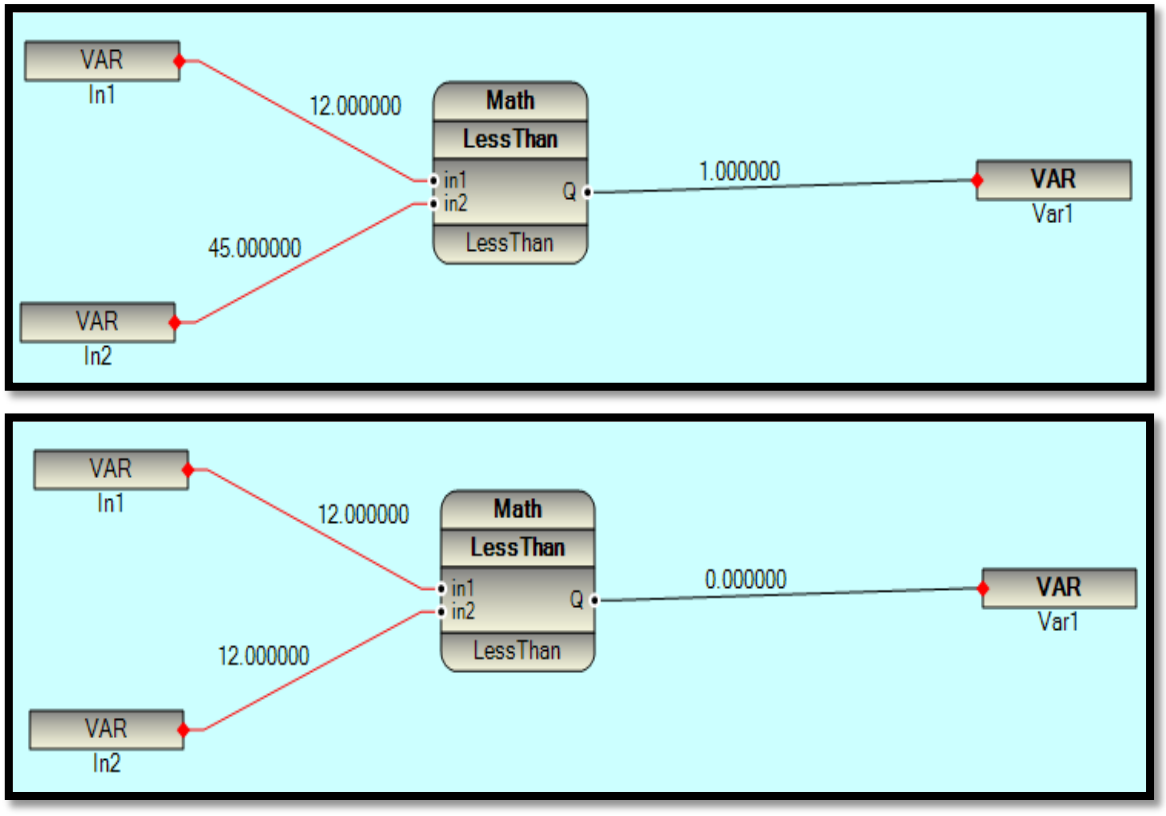
21.1.7 LessThan Function Block

LessThan Function Block accept two inputs (in1 and in2). If In1 is less Than to In2, Output is 1, otherwise Output is 0. The inputs and output numerical type is ***double****.*



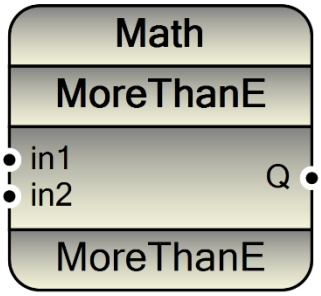
LessThan		
I/O	**Data Type**	**Description**
in1	**Double**	A
In2	**Double**	B
Q	**Double**	$If\ A < B \rightarrow Q=1\ If\ A \geq B \rightarrow Q=0$

For Example:



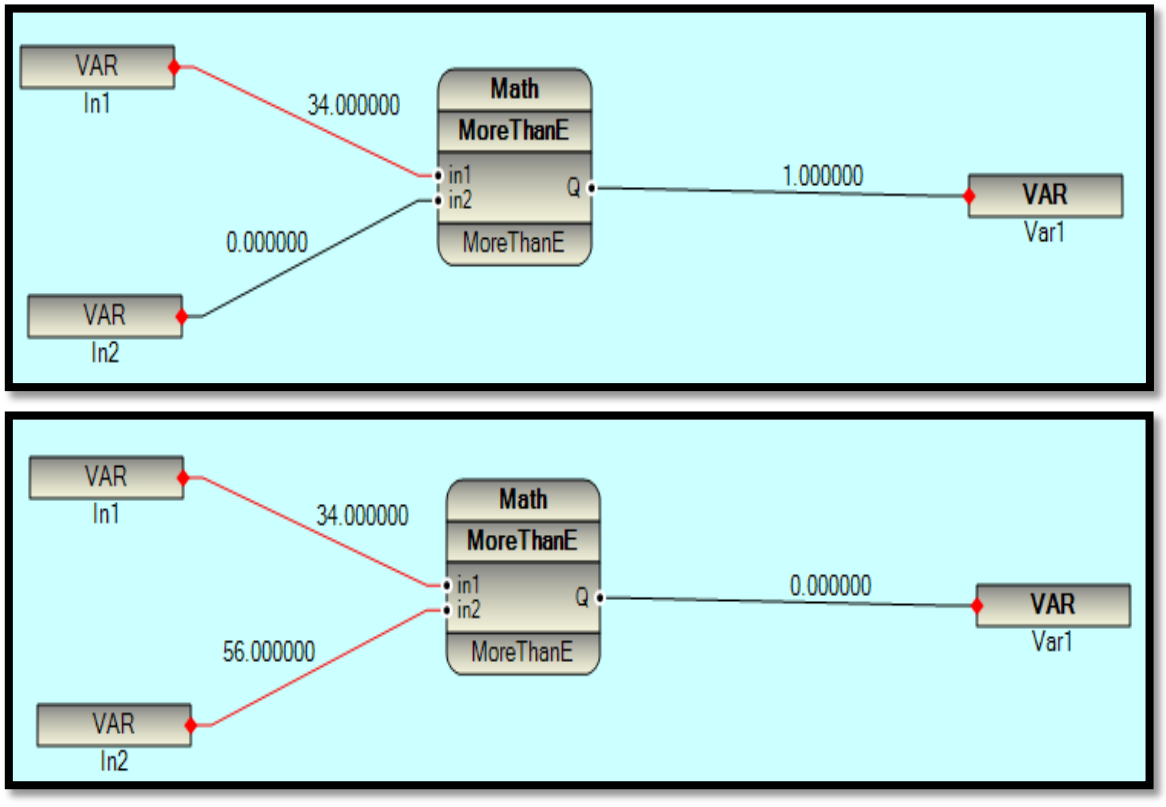
21.1.8 More Thane Function Block

MoreThanE Function Block accept two inputs (in1 and in2). If In1 is more Than or Equal to In2, Output is 1, otherwise Output is 0. The inputs and output numerical type is ***double****.*



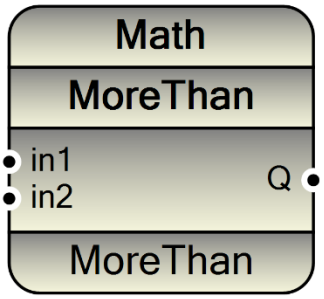
MoreThanE		
I/O	**Data Type**	**Description**
in1	**Double**	A
In2	**Double**	B
Q	**Double**	$If\ A \geq B \rightarrow Q=1\ If\ A < B \rightarrow Q=0$

For Example:



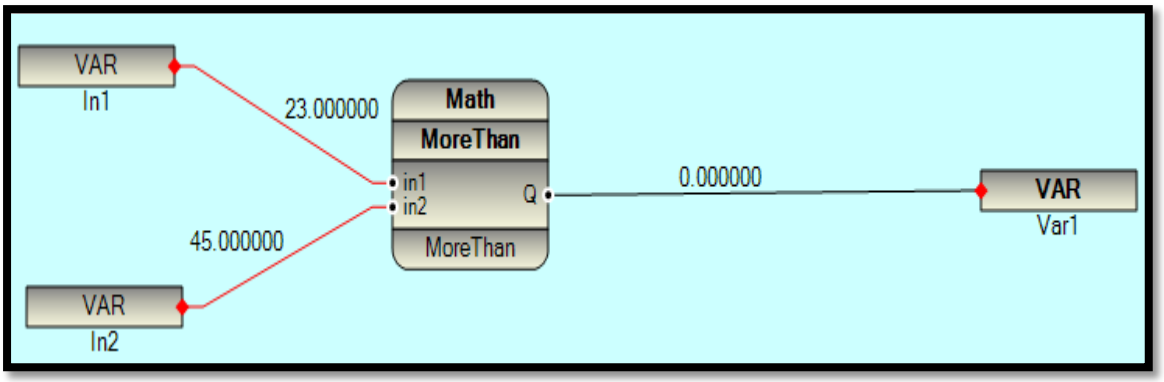
21.1.9 More Than Function Block

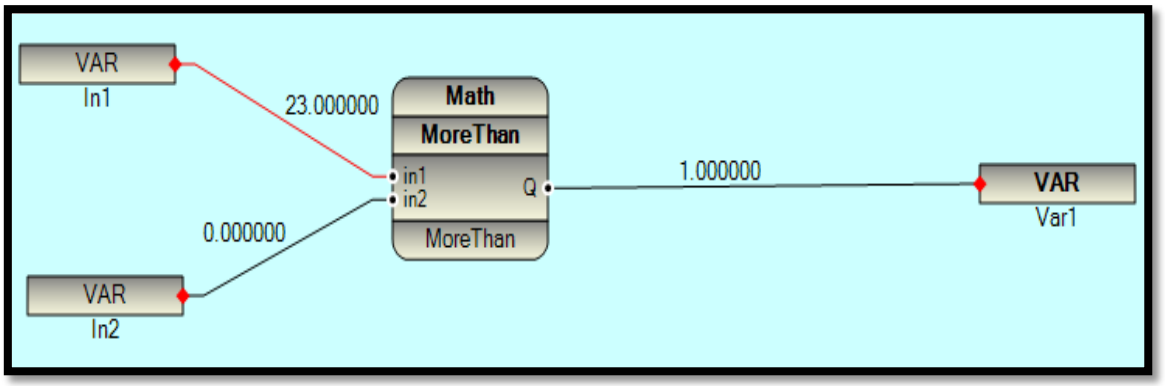
*MoreThan Function Block accept two inputs (in1 and in2). If In1 is more Than to In2, Output is 1, otherwise Output is 0. The inputs and output numerical type is **double**.*



MoreThan		
I/O	**Data Type**	**Description**
in1	**Double**	A
In2	**Double**	B
Q	**Double**	$If\ A > B \rightarrow Q=1\ If\ A \leq B \rightarrow Q=0$

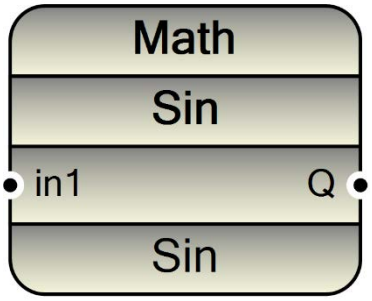
For Example:





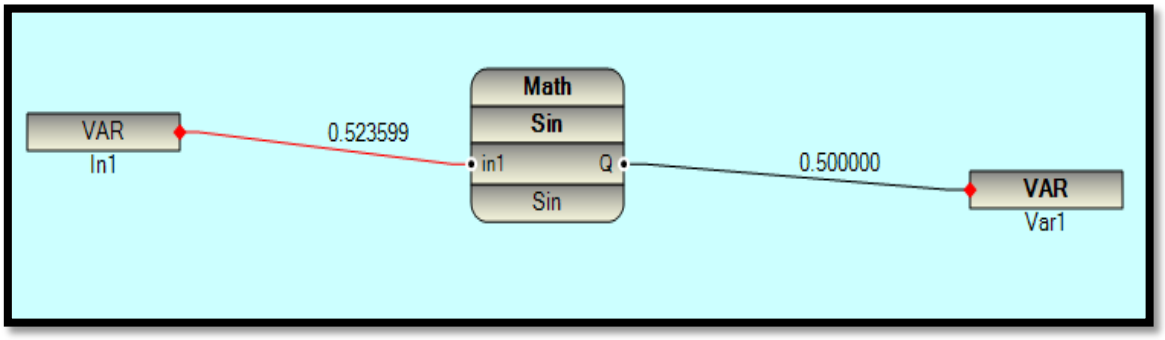
21.1.10 Sin Function Block

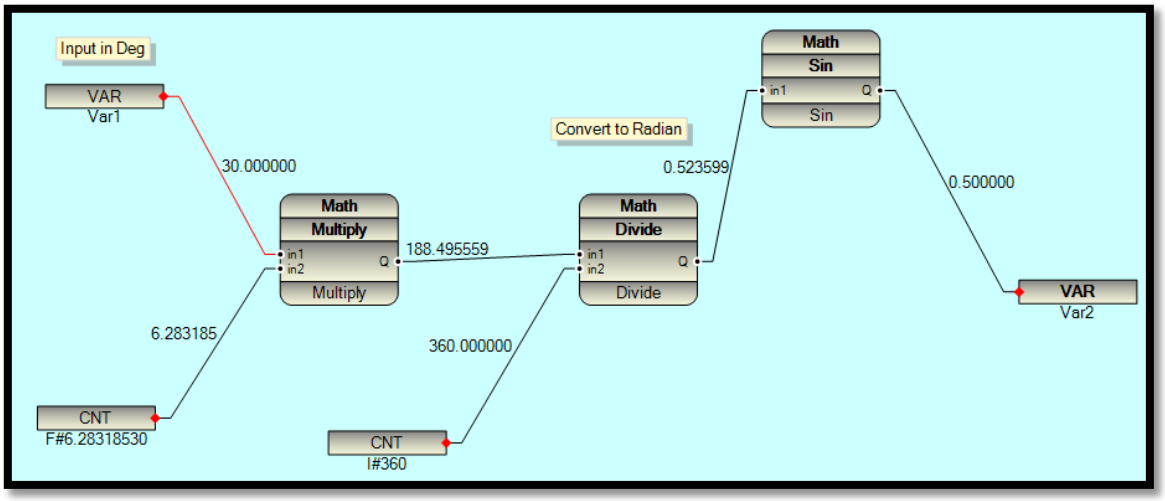
*Sin Function Block accept an input and returns **Sin(In1)**, where in is given in **Radians**.*



Sin		
I/O	**Data Type**	**Description**
in1	**Double**	*A(Radian)*
Q	**Double**	*Sin(A)*

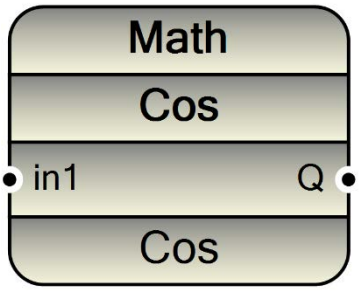
For Example:





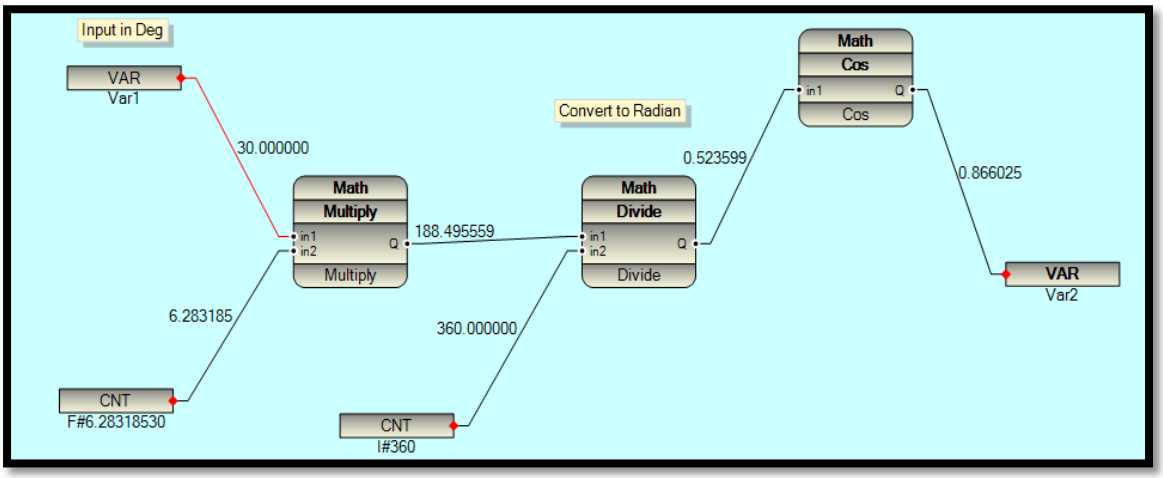
21.1.11 Cos Function Block

*cos Function Block accept an input and returns **cos(In1)**, where in is given in **Radians**.*



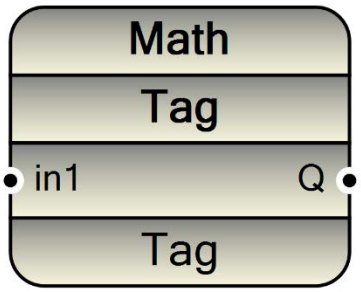
Cos		
I/O	**Data Type**	**Description**
in1	**Double**	*A(Radian)*
Q	**Double**	*Cos(A)*

For Example:



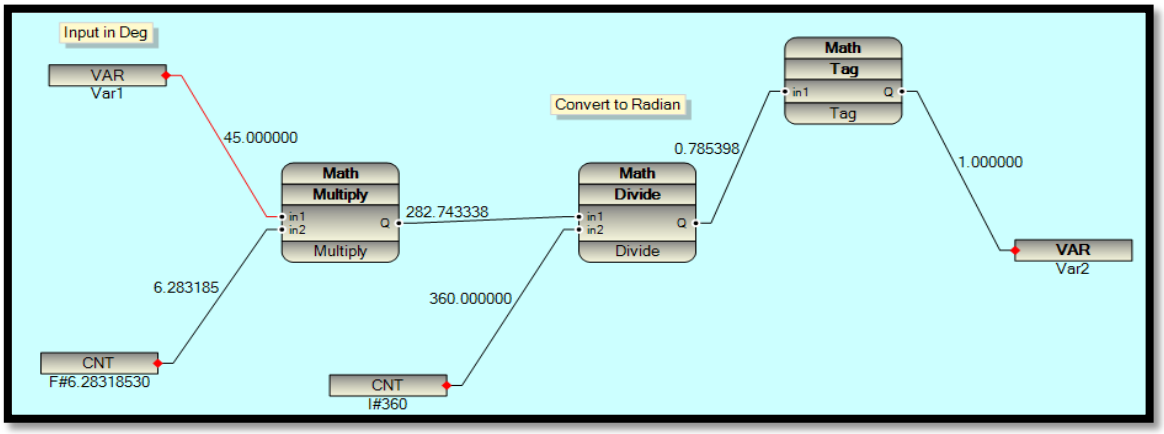
21.1.12 Tag Function Block

*Tag Function Block accept an input and returns T**ag(In1), where** in is given in **Radians**.*



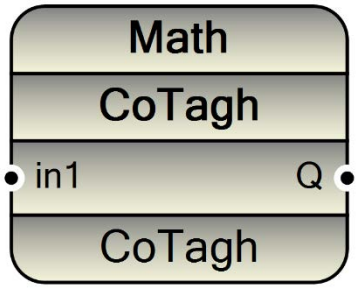
Tag		
I/O	**Data Type**	**Description**
in1	**Double**	*A(Radian)*
Q	**Double**	*Tag(A)*

For Example:



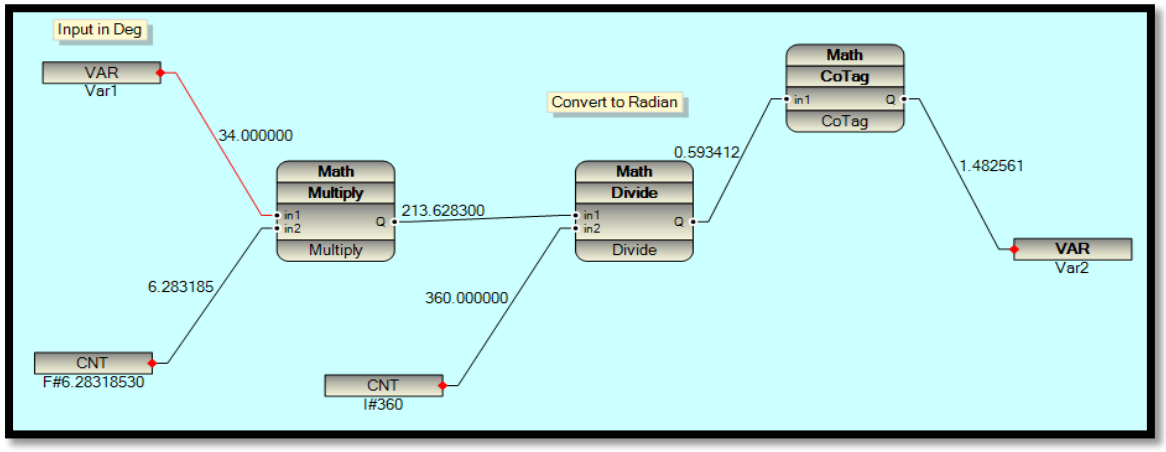
21.1.13 CoTag Function Block

*CoTag Function Block accept an input and returns **CoTag(In1)**, where in is given in **Radians**.*



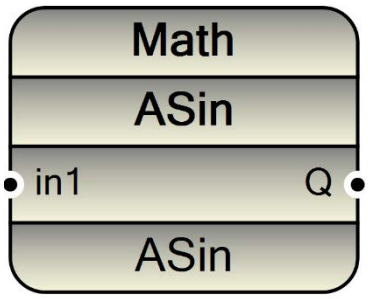
CoTag		
I/O	**Data Type**	**Description**
in1	**Double**	*A(Radian)*
Q	**Double**	*Cotag(A)*

For Example:



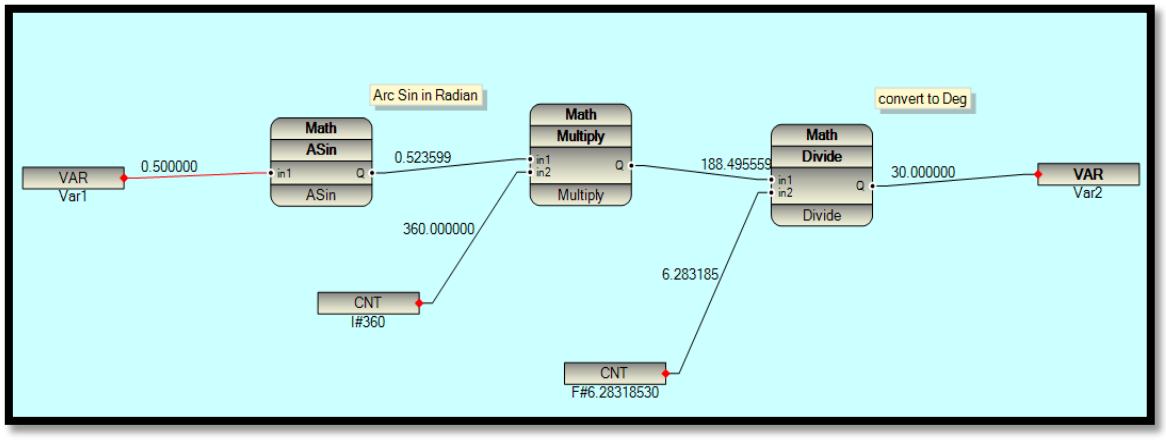
21.1.14 ASin Function Block

*ASin Function Block accept an input and returns **Arc sin(In1), where** in is given in **Radians**.*



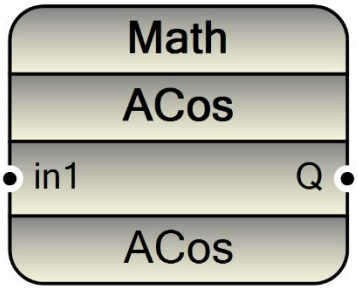
ASin		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc sin(A) (Radian)*

For Example:



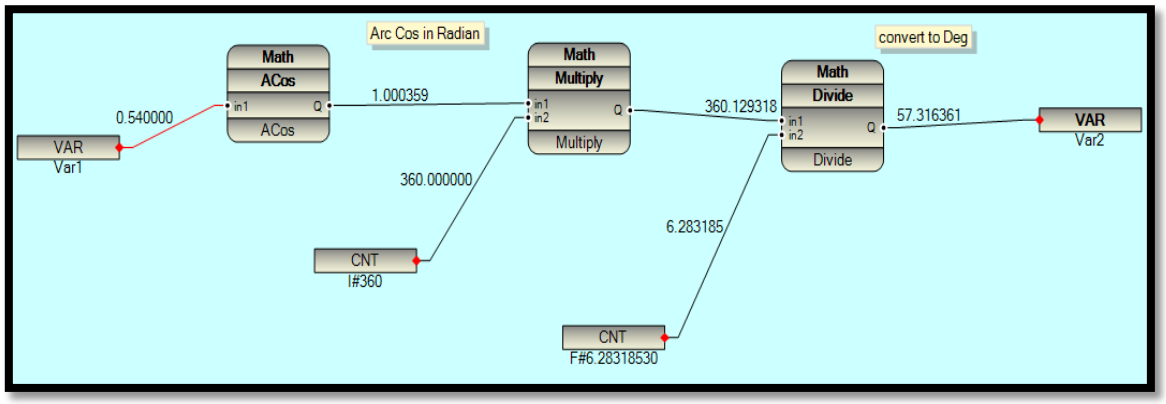
21.1.15 ACos Function Block

*ACos Function Block accept an input and returns **Arc cos(In1)**, where in is given in **Radians**.*



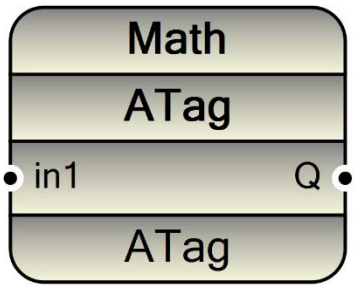
ACos		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc cos(A) (Radian)*

For Example:



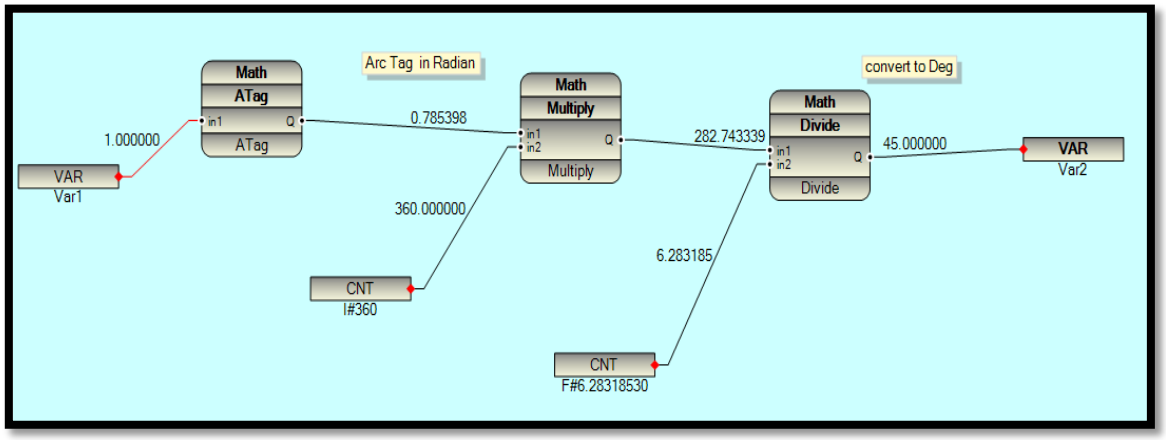
21.1.16 ATag Function Block

*ATag Function Block accept an input and returns **Arc tag(In1)**, where in is given in **Radians**.*



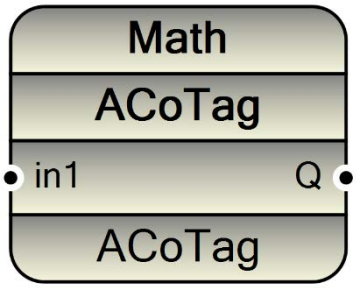
ATag		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc tag(A) (Radian)*

For Example:



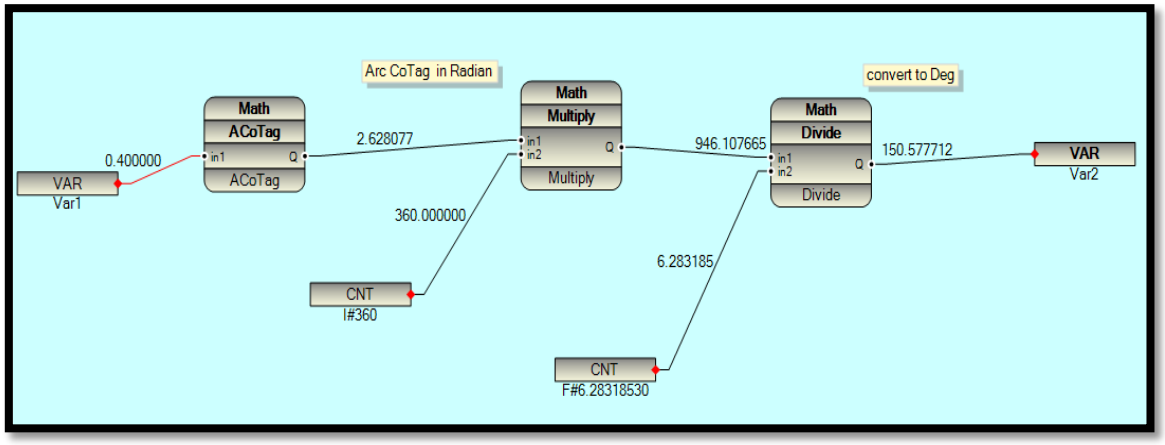
21.1.17 ACoTag Function Block

ACoTag Function Block accept an input and returns ***Arc cotag(In1)****, where in is given in* ***Radians****.*



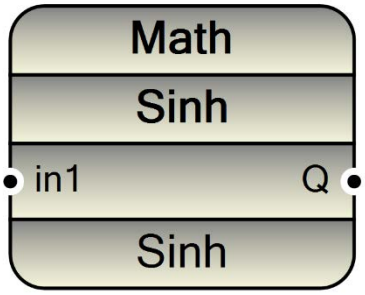
ACotag		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc cotag(A) (Radian)*

For Example:



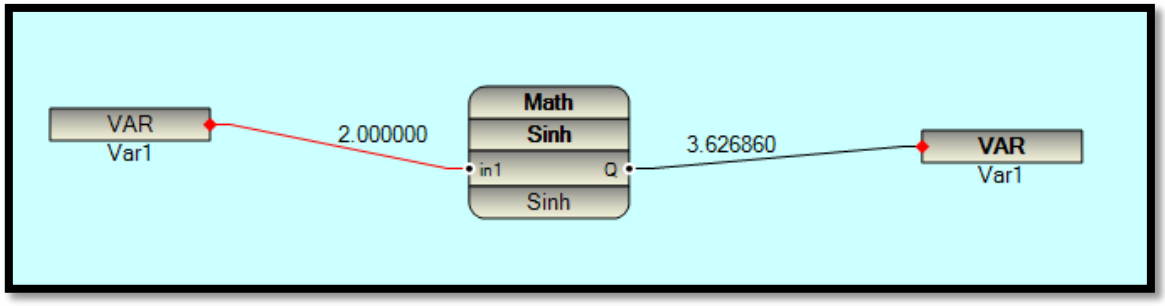
21.1.18 Sinh Function Block

Sinh Function Block accept a input and returns the ***hyperbolic sine of x****, which is defined mathematically as* $Sinh\ x = (exp(x) - exp(-x)) / 2$



Sinh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Sinh(A)*

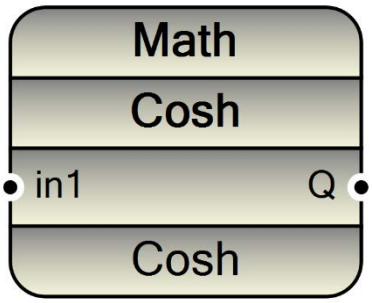
For Example: $Sinh(2) = 3.626860$



21.1.19 Cosh Function Block

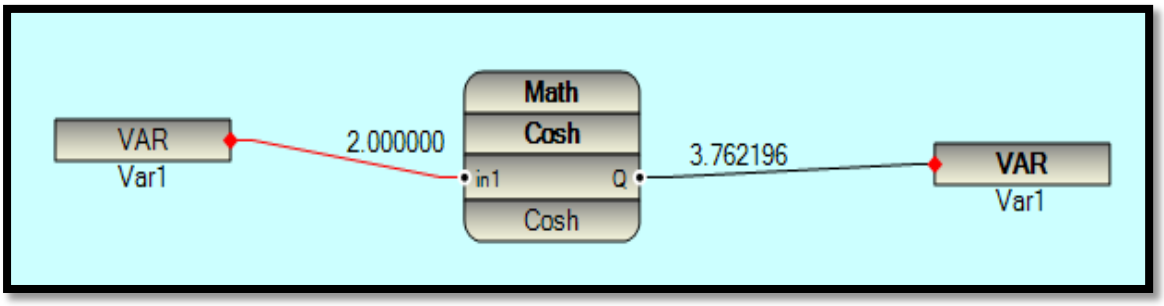
*Cosh Function Block accept a input and returns the **hyperbolic cosine of x**, which is defined mathematically as*

$cosh\, x = (exp(x) + exp(-x)) / 2$



Cosh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Cosh(A)*

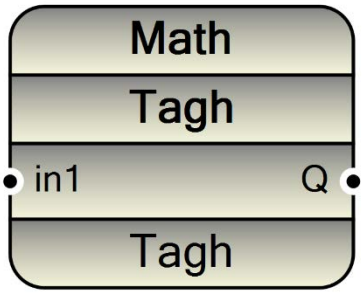
For Example: $Cosh(2) = 3.762196$



21.1.20 Tagh Function Block

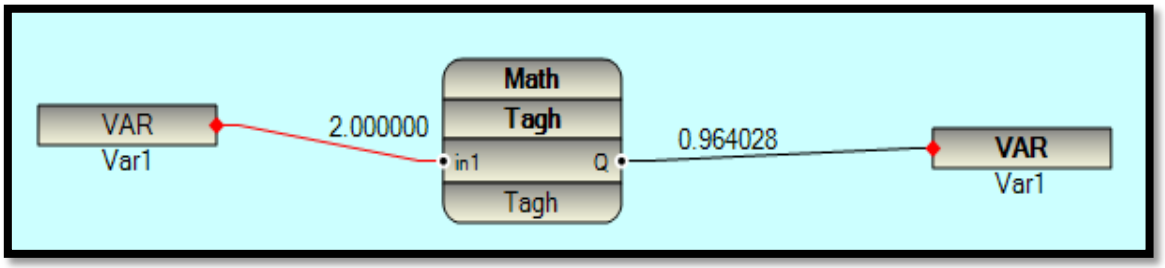
*Tagh Function Block accept a input and returns the **hyperbolic tagant of x**, which is defined mathematically as*

$Tagh\, x = \frac{Sinh\, x}{Cos\, x}$



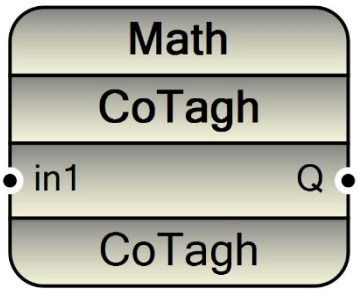
Tagh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Tagh(A)*

For Example: $Tagh(2) = 0.964028$



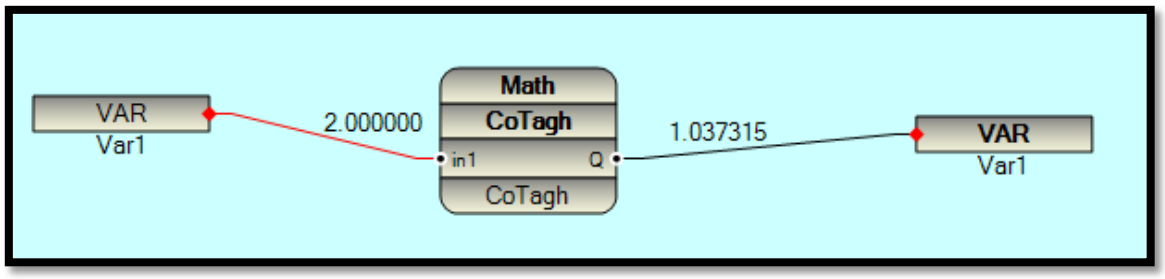
21.1.21 CoTagh Function Block

*CoTagh Function Block accept a input and returns the **hyperbolic cotagant of x**, which is defined mathematically as* $CoTagh\ x = \frac{Cosh\ x}{sinh\ x}$



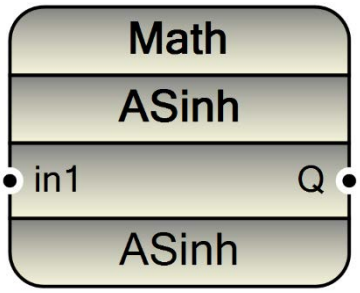
CoTagh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Cotagh(A)*

For Example: $Cotagh(2) = 1.037315$



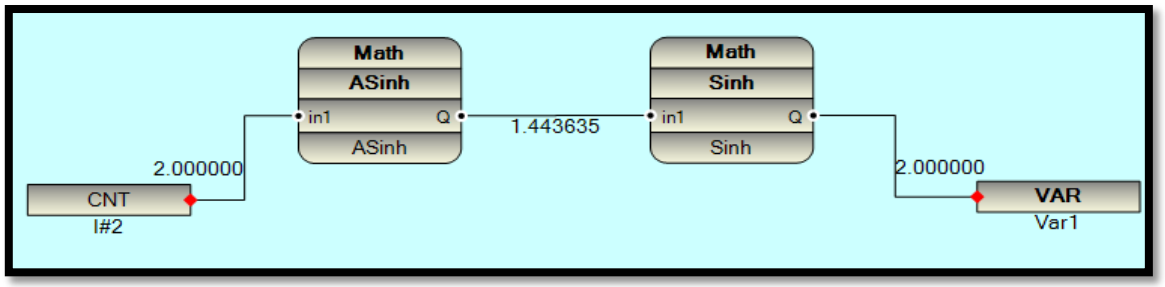
21.1.22 ASinh Function Block

*ASinh Function Block accept a input and returns the **Arc hyperbolic Sine of x**, which is defined mathematically as* $arcsinh\ x = ln(x + \sqrt{x^2 + 1}\ )$



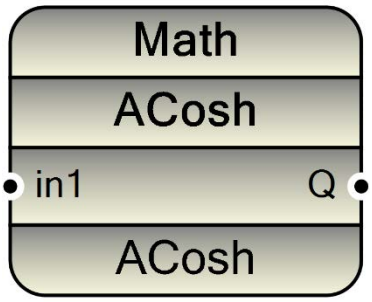
ASinh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc sinh(A)*

For Example: $Arc\ sinh(2) = 1.443635$



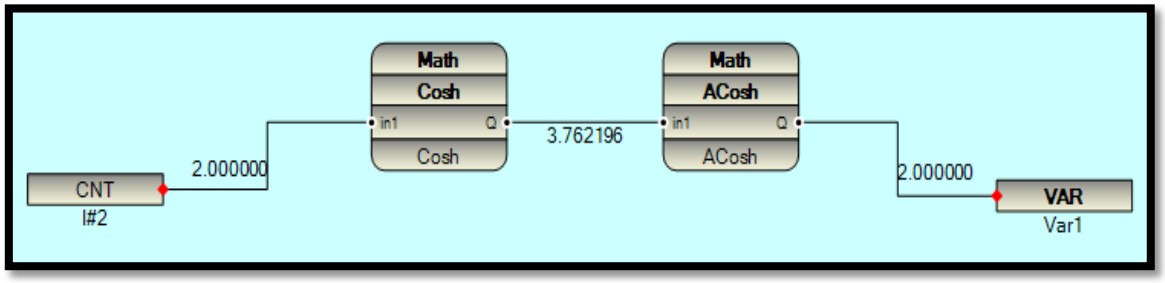
21.1.23 ACosh Function Block

ACosh Function Block accept a input and returns the ***Arc hyperbolic Cosine of x,*** *which is defined mathematically as* $arccosh\, x = ln(x + \sqrt{x^2 - 1})$



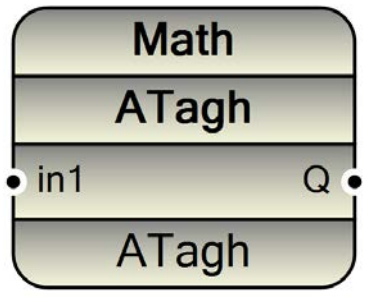
ACosh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc Cosh(A)*

For Example: $Arc\ cosh(3.762196) = 2$



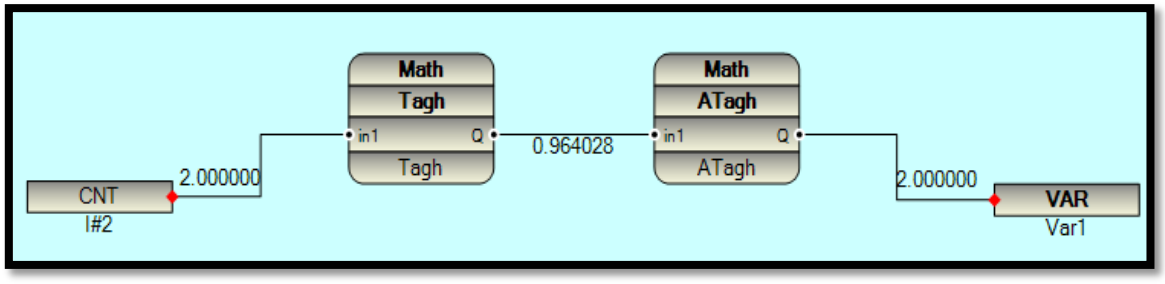
21.1.24 ATagh Function Block

ATagh Function Block accept a input and returns the ***Arc hyperbolic Tagant of x,*** *which is defined mathematically as* $arc\ tagh\, x = \frac{1}{2} ln\left(\frac{1+x}{1-x}\right)$



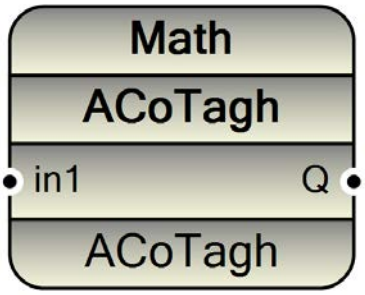
ATagh		
I/O	**Data Type**	**Description**
in1	**Double**	*A*
Q	**Double**	*Arc tagh(A)*

For Example: $Arc\ tagh(0.964028) = 2$



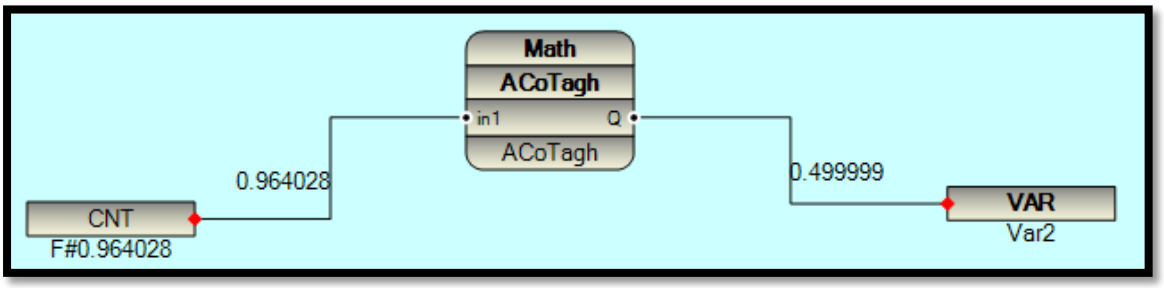
21.1.25 ACoTagh Function Block

ACoTagh Function Block accept a input and returns the ***Arc hyperbolic Cotag of x****, which is defined mathematically as* $arc\ cotagh\ x = \frac{1}{2} ln\left(\frac{x+1}{x-1}\right)$



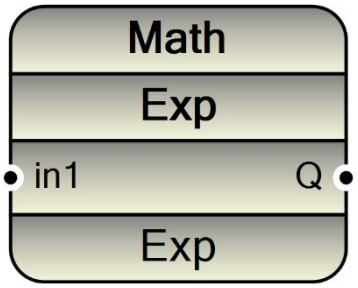
ACoTagh		
I/O	**Data Type**	**Description**
in1	**Double**	A
Q	**Double**	$Arc\ cotagh(A)$

For Example: $Arc\ cotagh(0.964028) \cong 0.5$



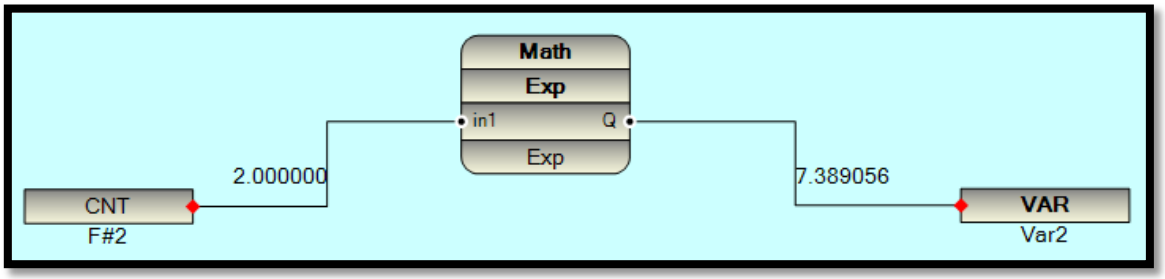
21.1.26 Exp Function Block

Exp Function Block accept a input and returns $\boldsymbol{e^x}$



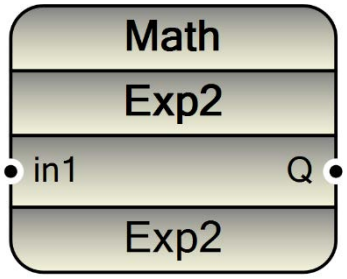
Exp		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	e^x

For Example: $e^2 = 7.389056$



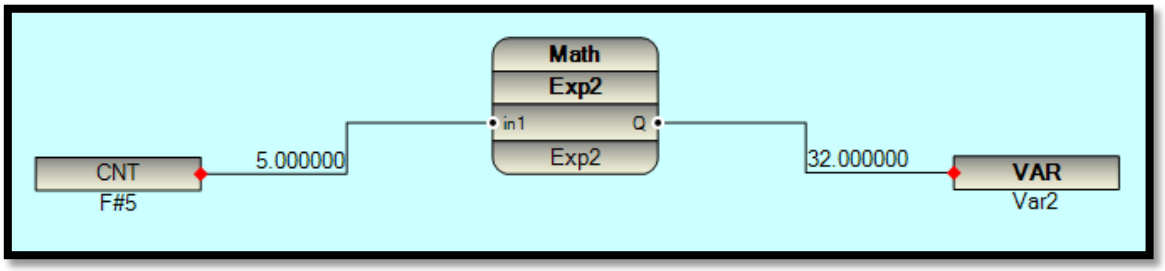
21.1.27 Exp2 Function Block

Exp2 Function Block accept a input and returns $\mathbf{2^x}$



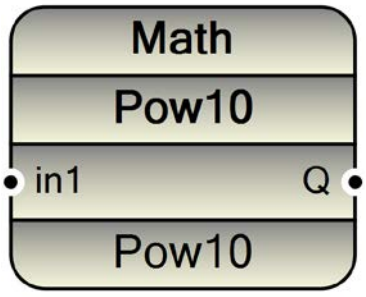
Exp2		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	2^x

For Example: $2^5 = 32$



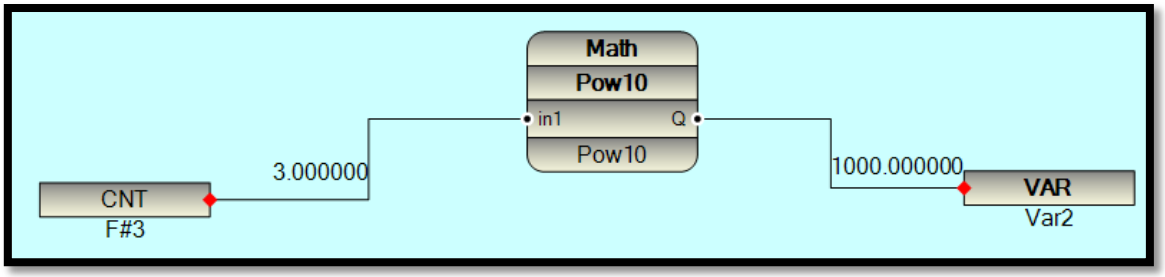
21.1.28 Pow10 Function Block

Pow10 Function Block accept a input and returns $\mathbf{10^x}$



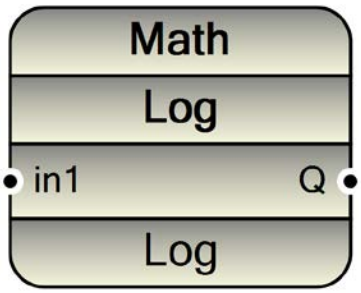
Pow10		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	10^x

For Example: $10^3 = 1000$



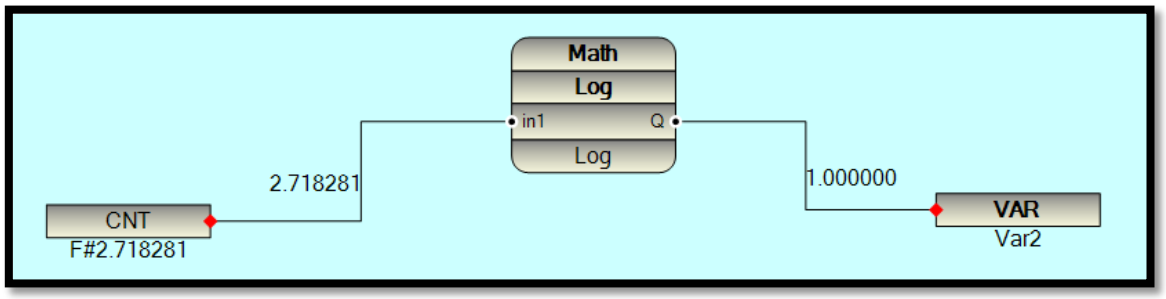
21.1.29 Log Function Block

Log Function Block accept a input and returns $\boldsymbol{log_e^x}$ *,which is defined mathematically as* $ln(x)$



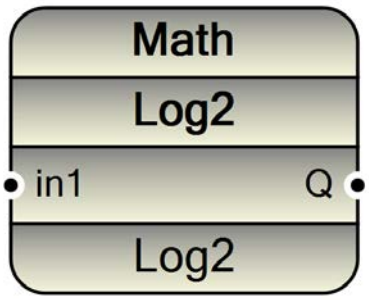
Log		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\ln(x)$

For Example: $log_e^{2.718281} = ln(e) = 1$



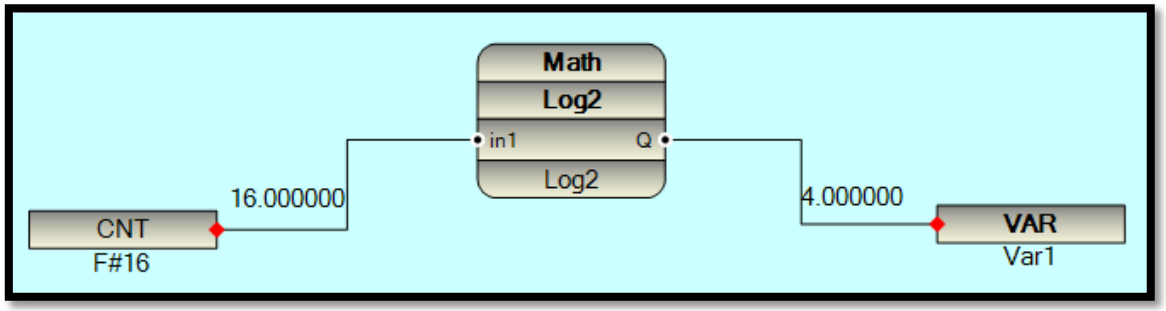
21.1.30 Log2 Function Block

Log2 Function Block accept a input and returns $\boldsymbol{log_2^x}$



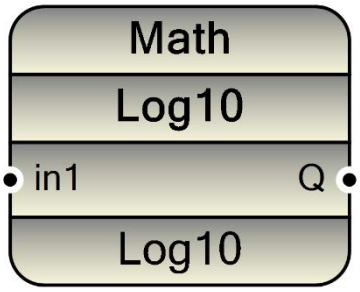
Log2		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	log_2^x

For Example: $log_2^{16} = 4$



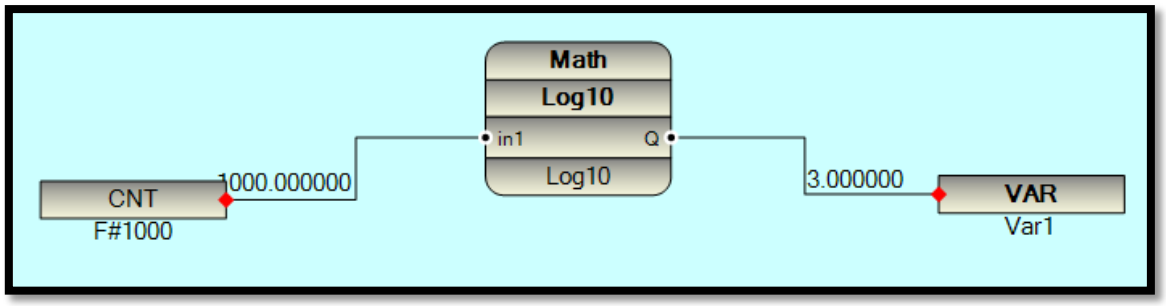
21.1.31 Log10 Function Block

Log10 Function Block accept a input and returns $\boldsymbol{log_{10}^{x}}$



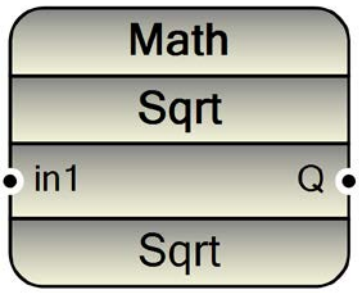
Log10		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	log_{10}^{x}

For Example: $log_{10}^{1000} = 3$



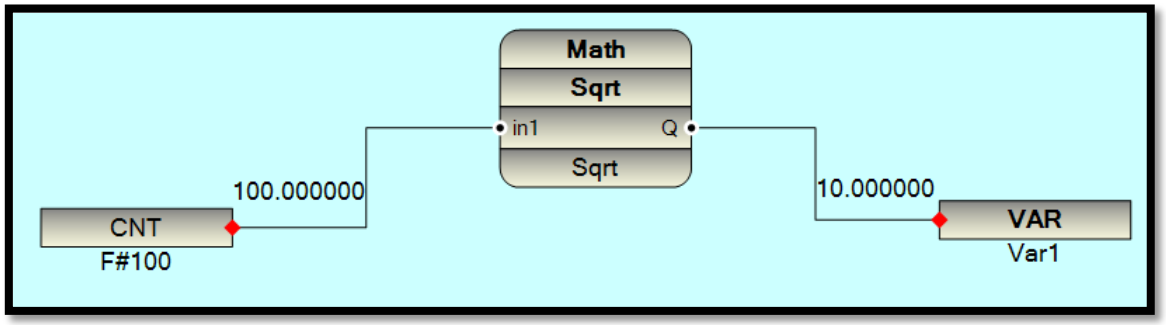
21.1.32 Sqrt Function Block

Sqrt Function Block accept a input ,and returns ***square root of*** $\boldsymbol{x}$, $\boldsymbol{Sqrt(x)}$ *,which is defined mathematically as* $\sqrt[2]{x}$



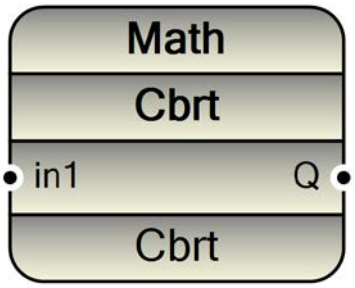
Sqrt		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\sqrt[2]{x}$

For Example: $Sqrt(100) = \sqrt{100} = 10$



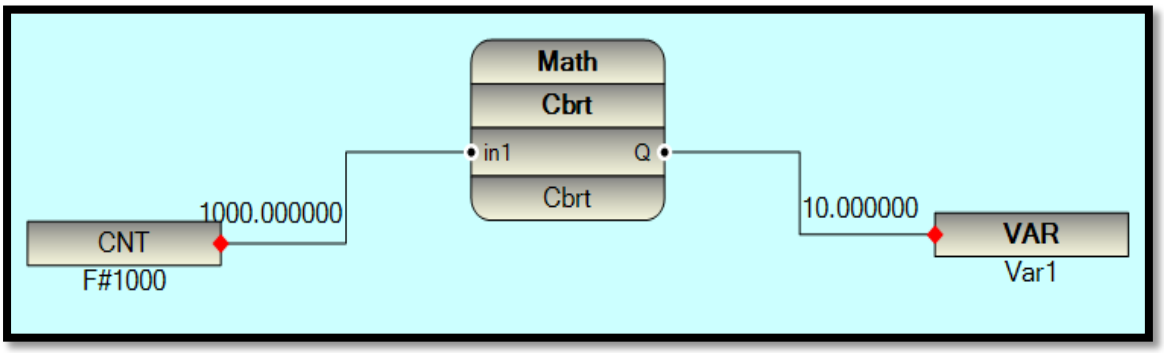
21.1.33 Cbrt Function Block

Cbrt Function Block accept a input ,and returns ***Cube root of x, Cbrt(x)*** *,which is defined mathematically as* $\sqrt[3]{x}$



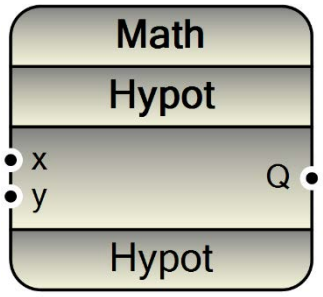
Cbrt		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\sqrt[3]{x}$

For Example: $Sqrt(1000) = \sqrt{1000} = 10$



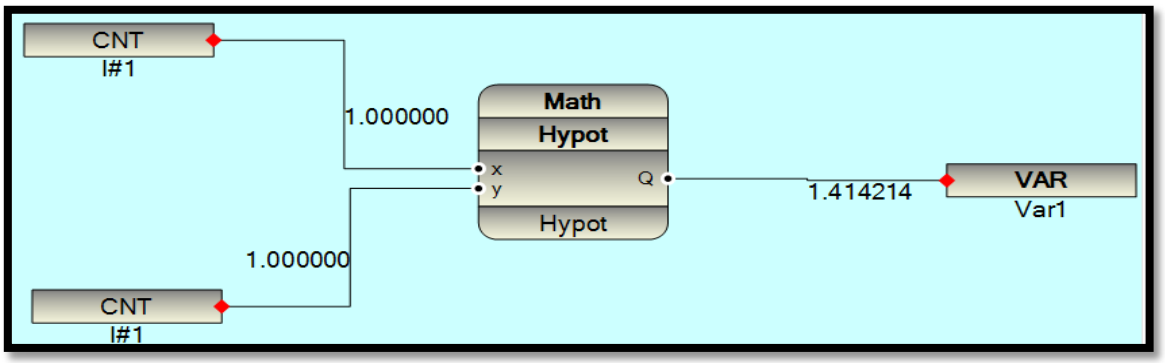
21.1.34 Hypot Function Block

Hypot Function Block accept a input ,and returns ***hypotenuse****, which is defined mathematically as* $\sqrt{x^2 + y^2}$



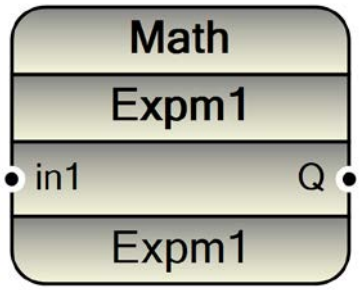
Hypot		
I/O	**Data Type**	**Description**
x	**Double**	x
y	**Double**	y
Q	**Double**	$\sqrt{x^2 + y^2}$

For Example: $Hypot(1,1) = \sqrt{1^2 + 1^2} = 1.414214$



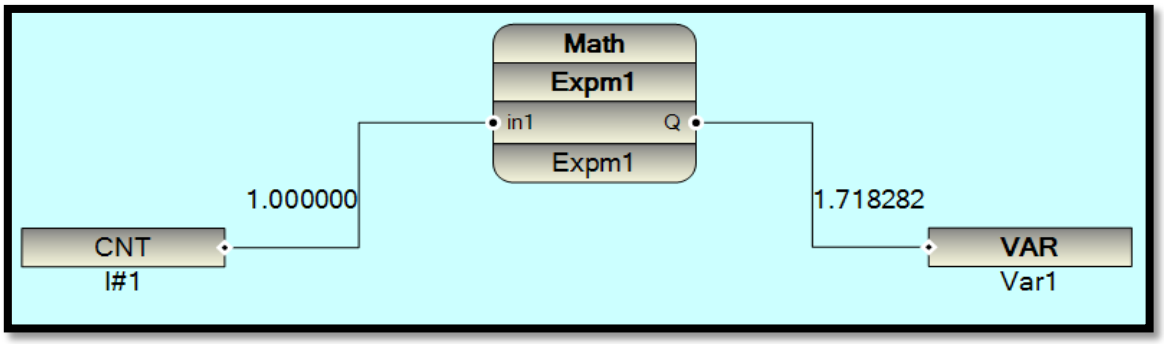
21.1.35 Expm1 Function Block

Expm1 Function Block accept a input ,and returns $e^x - 1$



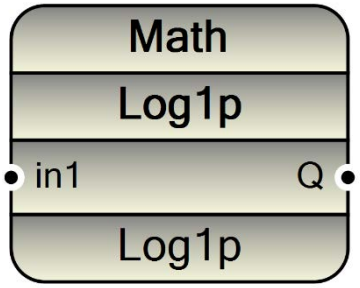
Hypot		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$e^x - 1$

For Example: $Expm(1) = e^1 - 1 = 1.718282$



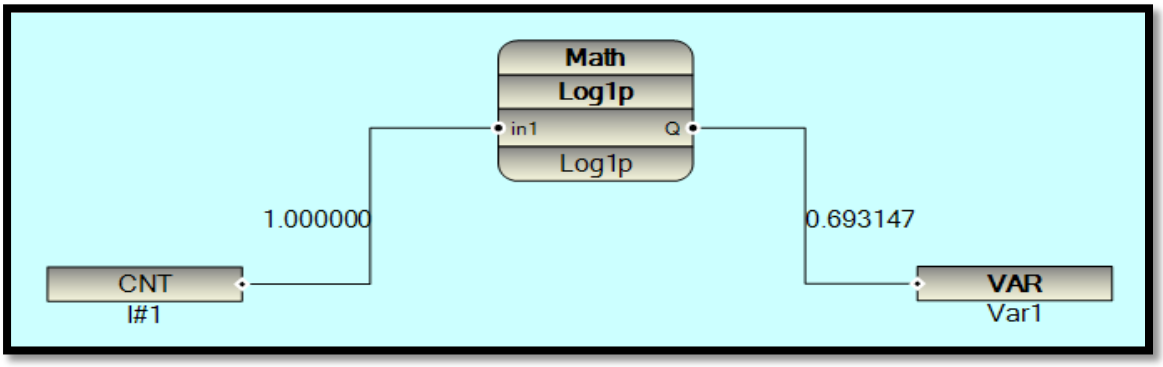
21.1.36 Log1p Function Block

Log1p Function Block accept a input ,and returns $\ln(x + 1)$



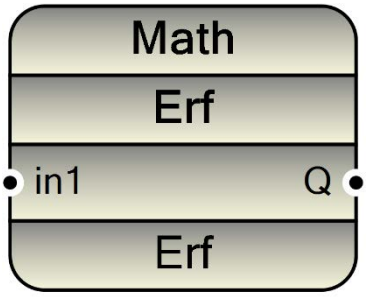
Log1p		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\ln(1 + x)$

For Example: $Log1p(1) = \ln(1 + 1) = 0.693147$



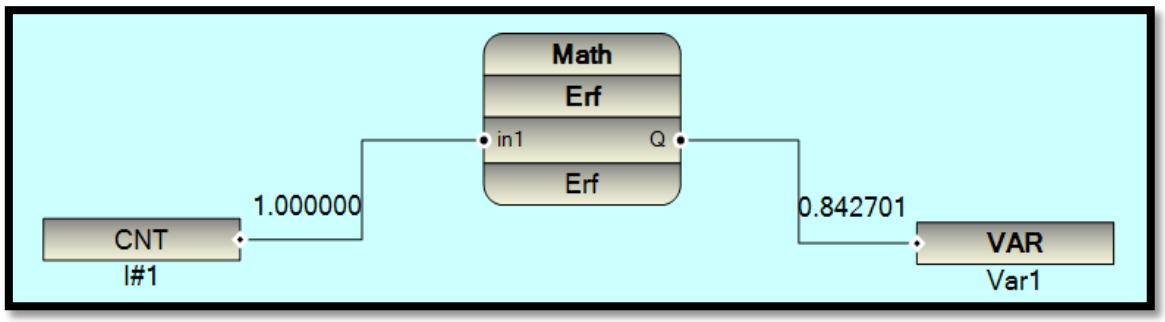
21.1.37 Erf Function Block

Erf Function Block accept a input ,and returns ***Error Function****, which is defined mathematically as* $\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$



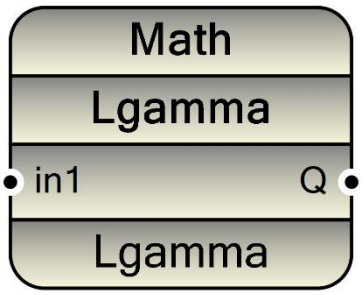
Erf		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$

For Example: $Erf(1)=\frac{2}{\sqrt{\pi}}\int_0^1 e^{-t^2}dt = 0.842701$



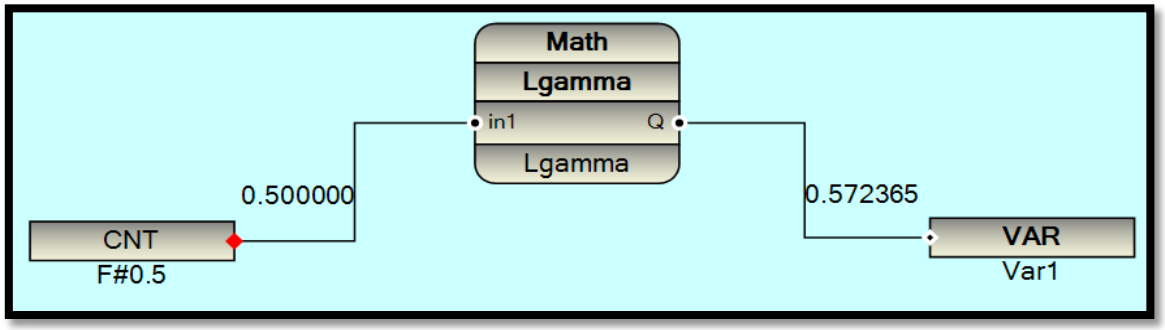
21.1.38 Lgamma Function Block

Lgamma Function Block accept a input ,and returns ***Log-gamma function****, which is defined mathematically as* $log_e^{|\Gamma(x)|}$



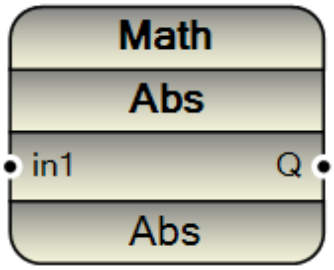
Lgamma		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$log_e^{\lvert\Gamma(x)\rvert}$

For Example: $Lgamma(0.5) = log_e^{|\Gamma(0.5)|} = 0.572365$



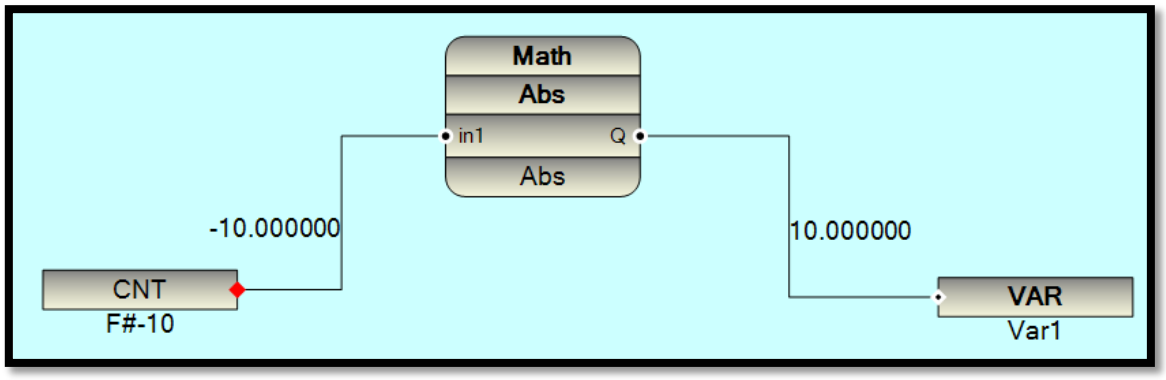
21.1.39 Abs Function Block

Abs Function Block accept a input ,and returns ***Absolut value****, which is defined mathematically as* $|x|$



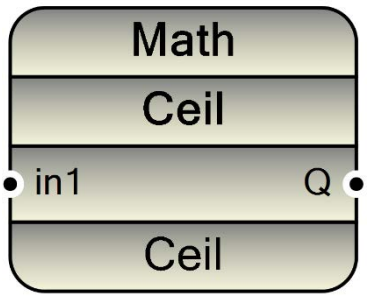
Abs		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\|x\|$

For Example: $Abs(-10) = 10$



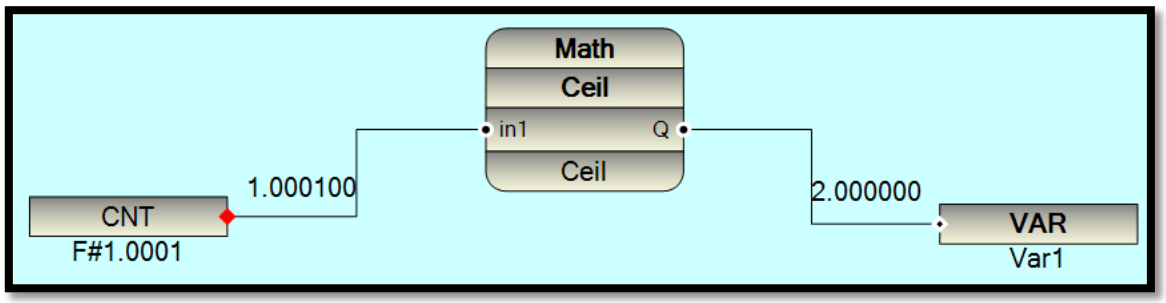
21.1.40 Ceil Function Block

Ceil Function Block accept a input ,and returns ***ceiling function****.the ceil function always round a number up to the next largest whole number. which is defined mathematically as* $\lceil x \rceil$



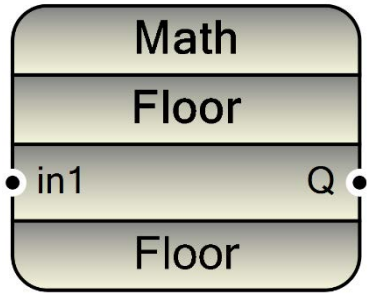
Ceil		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\lceil x \rceil$

For Example: $ceil(1.0001) = \lceil 1.0001 \rceil = 2$



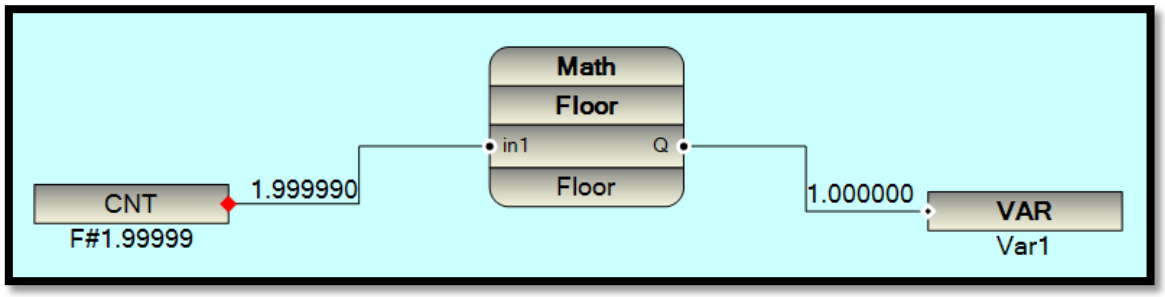
21.1.41 Floor Function Block

Floor Function Block accept a input ,and returns ***Flooring function****, floor function returns the largest integer less than or equal a given number.which is defined mathematically as* $\lfloor x \rfloor$



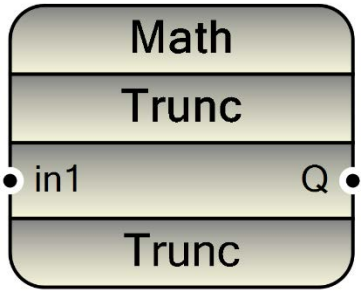
Floor		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$\lfloor x \rfloor$

For Example: $Floor(1.99999) = \lfloor 1.99999 \rfloor = 1$



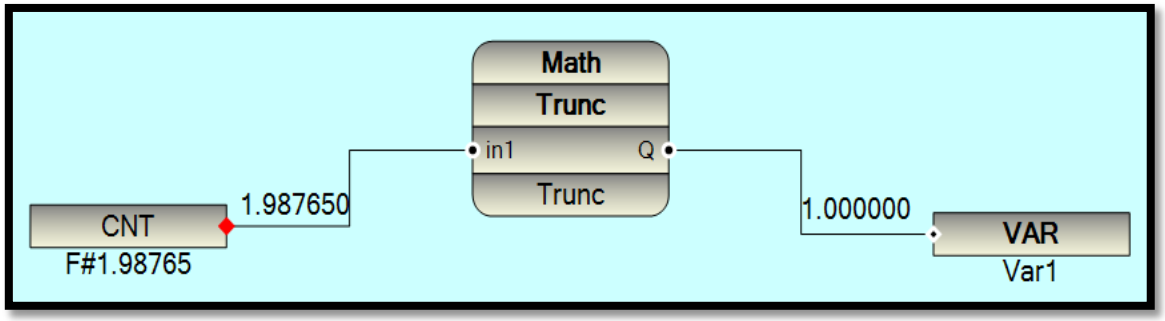
21.1.42 Trunc Function Block

Trunc Function Block accept a input ,and returns ***Trancation function*** *.*
the Trunc function return the integer part of a number by removing any fractional digits. which is defined mathematically as $[x]$



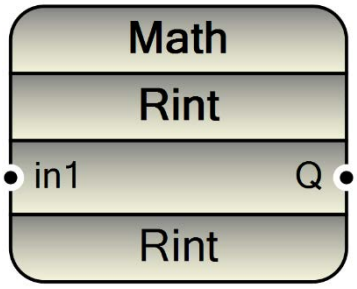
Trunc		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$[x]$

For Example: $Trunc(1.98765) = [1.98765] = 1$



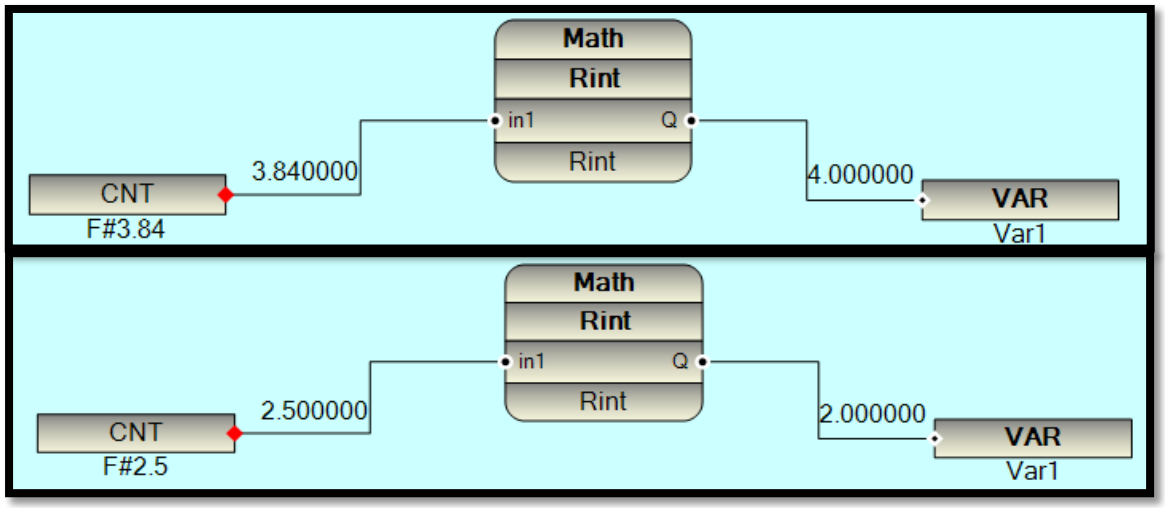
21.1.43 Rint Function Block

Rint Function Block accept a input ,and returns ***the doble value that is closest in value to the argument and is equal to a mathematical integer.***



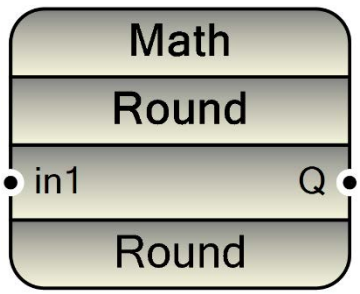
Rint		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$if\ x-[x]\geq 0.5 \rightarrow [x]+1$ $if\ x-[x]<0.5 \rightarrow [x]$

For Example: $Rint(3.84)=4$



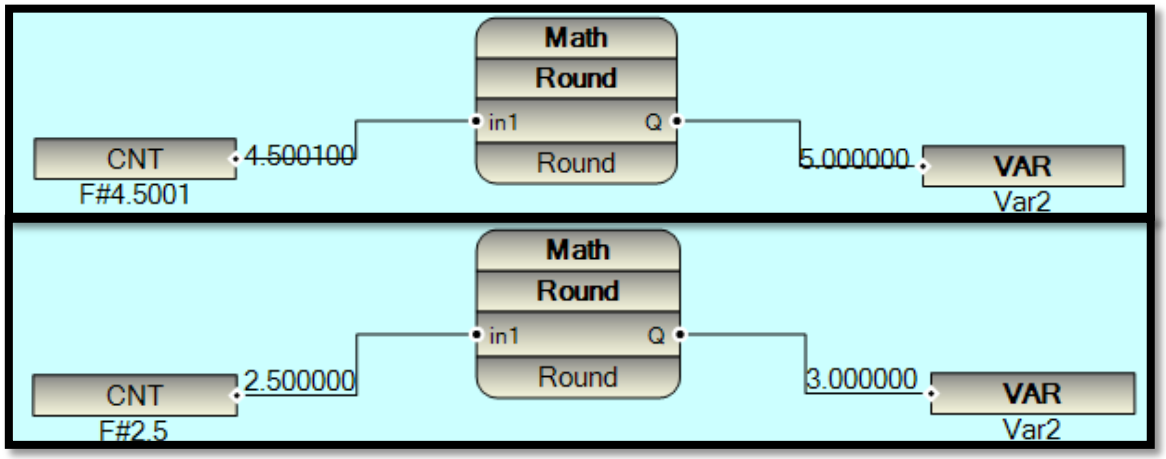
21.1.44 Round Function Block

Round Function Block accept a input ,and returns ***the value of a number rounded to the nearest integer .round returns a whole number of int/long.***



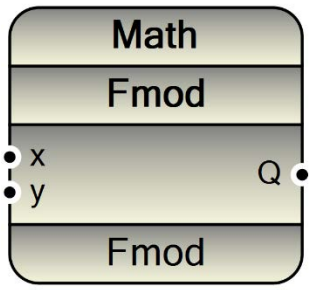
Round		
I/O	**Data Type**	**Description**
in1	**Double**	x
Q	**Double**	$if\ x-[x]\geq 0.5 \rightarrow [x]+1$ $if\ x-[x]<0.5 \rightarrow [x]$

For Example: $Round(4.5001)=5$



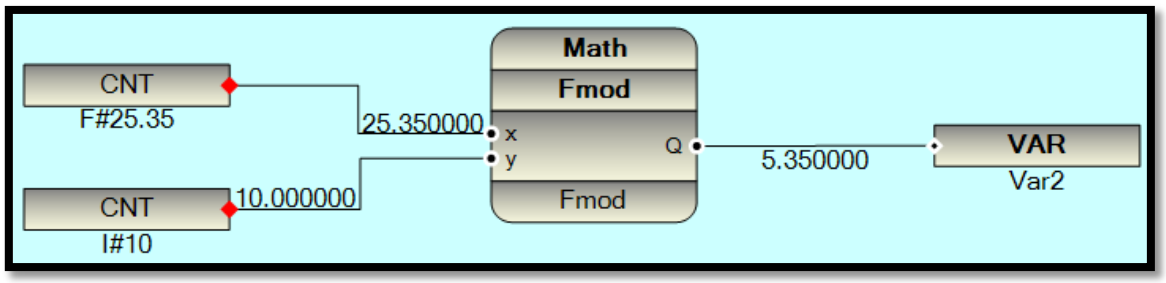
21.1.45 Fmod Function Block

Round Function Block accept two input (x,y) ,and ***computes floating-point remainder of the devision operation*** $\left({}^{x}/_{y}\right)$



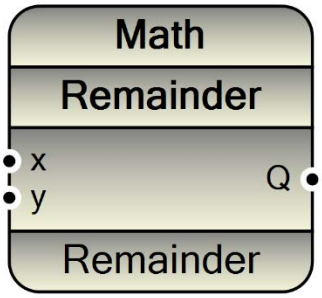
Fmod		
I/O	**Data Type**	**Description**
x	**Double**	x
y	**Double**	y
Q	**Double**	$x - ([{}^{x}/_{y}] \times y)$

For Example: $Fmod(25.35,20) = 25.35 - \left(\left[{}^{25.35}/_{10}\right] * 10\right) = 5.35$



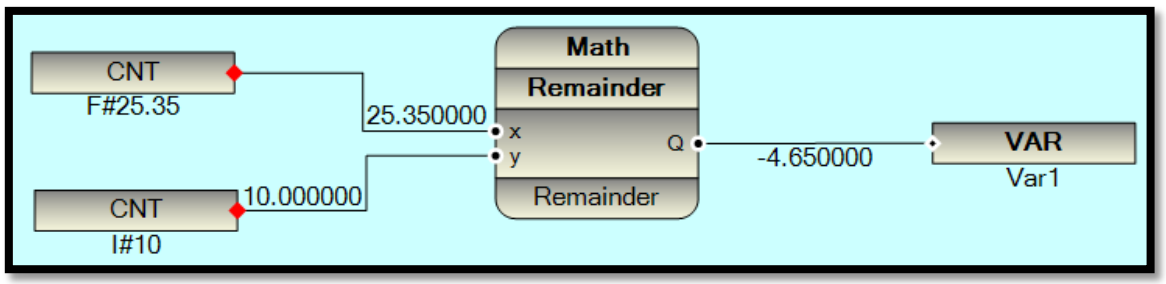
21.1.46 Remainder Function Block

Remainder Function Block accept two input (x,y) ,and ***returns the floating-point remainder of numer/denom (round to nearest)***



Remainder		
I/O	**Data Type**	**Description**
x	**Double**	x
y	**Double**	y
Q	**Double**	$x - (Round({}^{x}/_{y}) \times y)$

For Example: $Fmod(25,20) = 25.35 - \left(Round\left({}^{25.35}/_{10}\right) * 10\right) = -4.65$



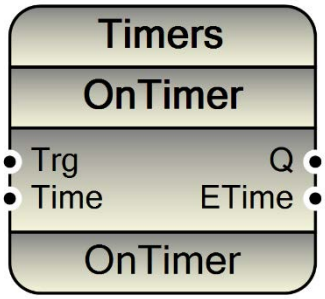
21.2 Timer Group:

Timer Function Groups

21.2.1 OnTimer Function Block

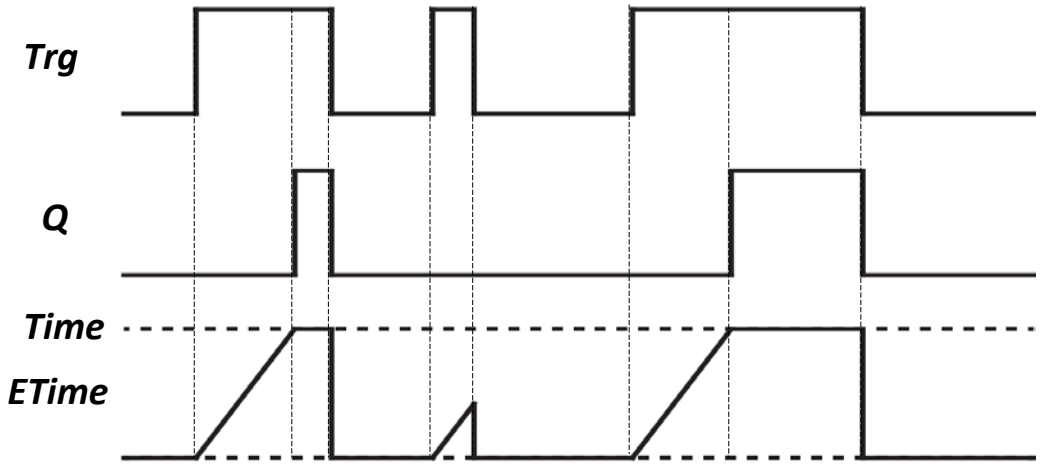
When Trg Input Change from 0 to 1 , After passing Time Input (in msec) , Output Q will changed to 1 . Etime Output Shows Time passed in Msec . If Trg Input Changed from 1 to 0, Time will reset. (Q and ETime Output change to 0)

OnTimer

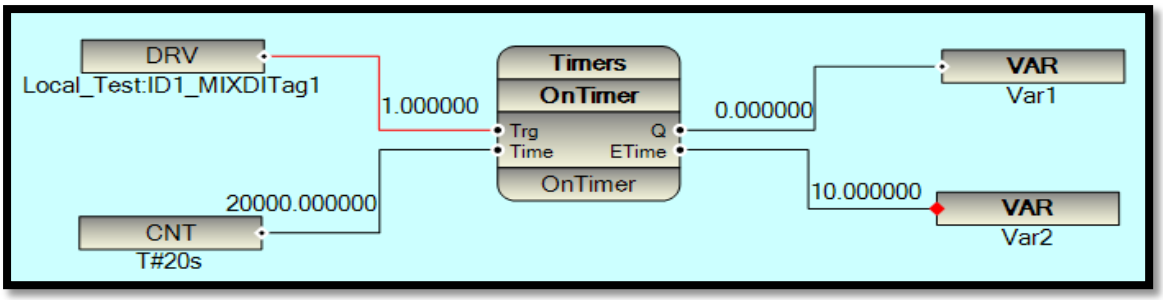


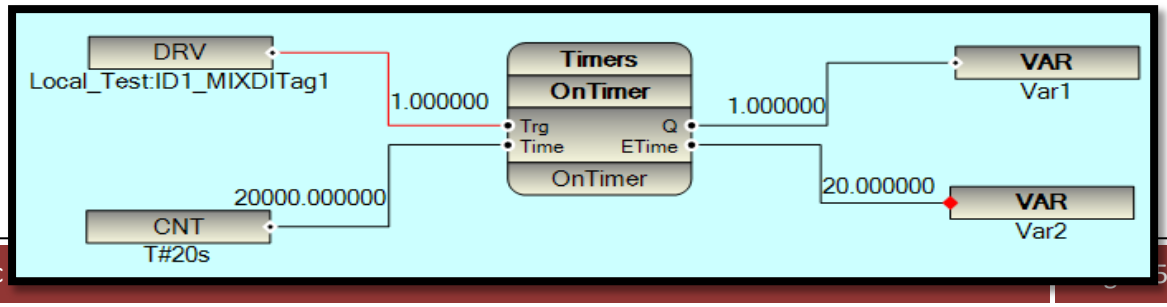
I/O	Data Type	Description
Trg	Double	*Timer operation condition*
Time	Double	*Preset Time*
Q	Double	*Timer output*
ETimer	Double	*Elapsed Time*

Timing Diagram



For Example:

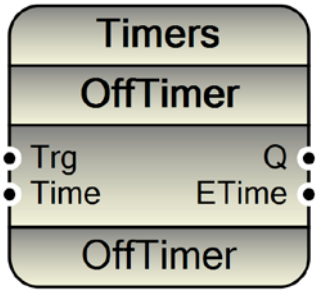




21.2.2 OffTimer Function Block

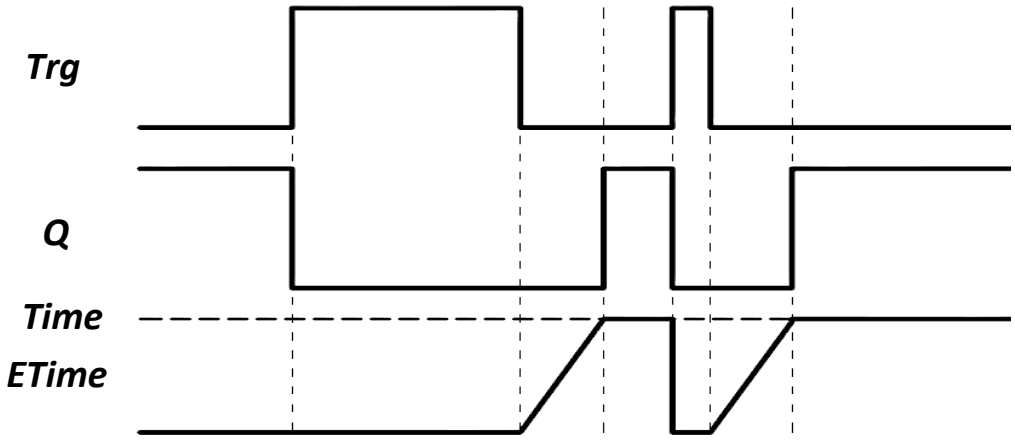
When Trg Input Change from 0 to 1 , Time will armed (Q = 0 , ETime = 0) and wait for detecting Trg Falling from 1 to 0 . When Trg Changed from 1 to 0 , Timer will start and after Time Msec , Q will change to 1 . ETime shows Passed Time in Msec .

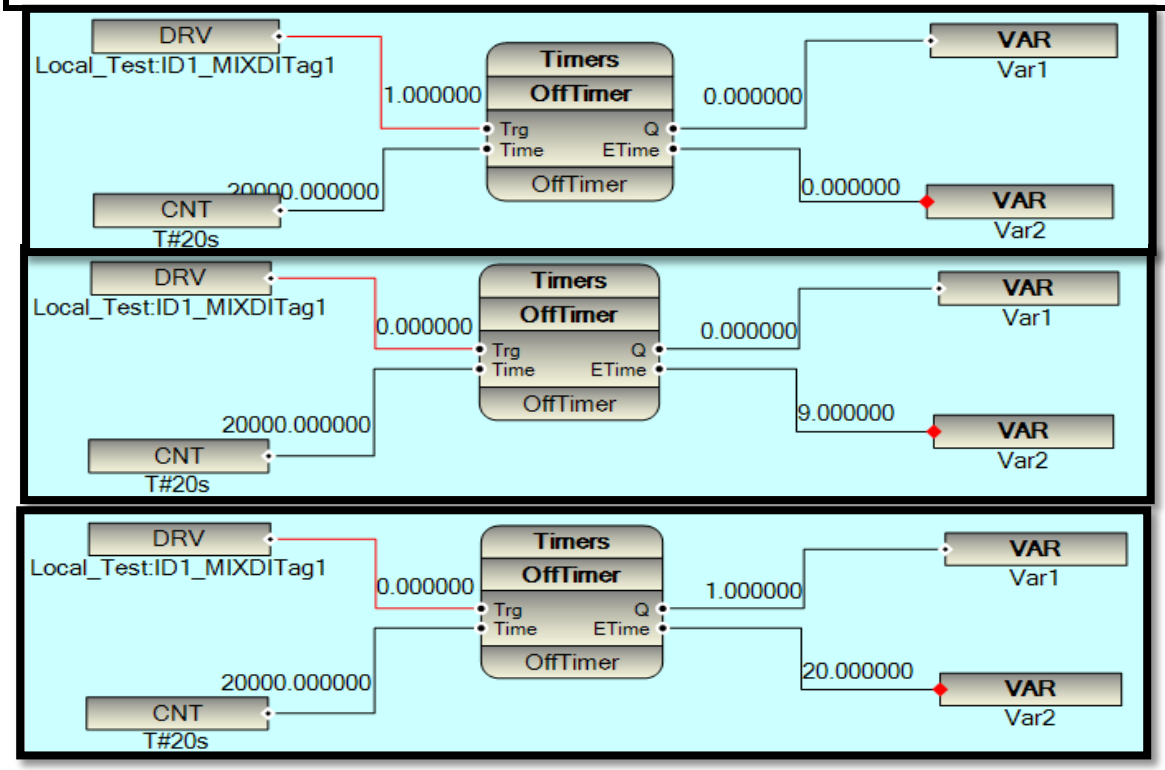
OnTimer



I/O	Data Type	Description
Trg	Double	*Timer operation condition*
Time	Double	*Preset Time*
Q	Double	*Timer output*
ETimer	Double	*Elapsed Time*

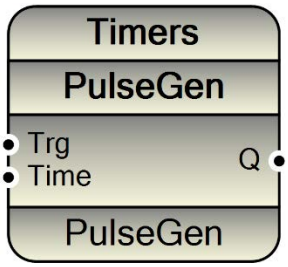
Timing Diagram



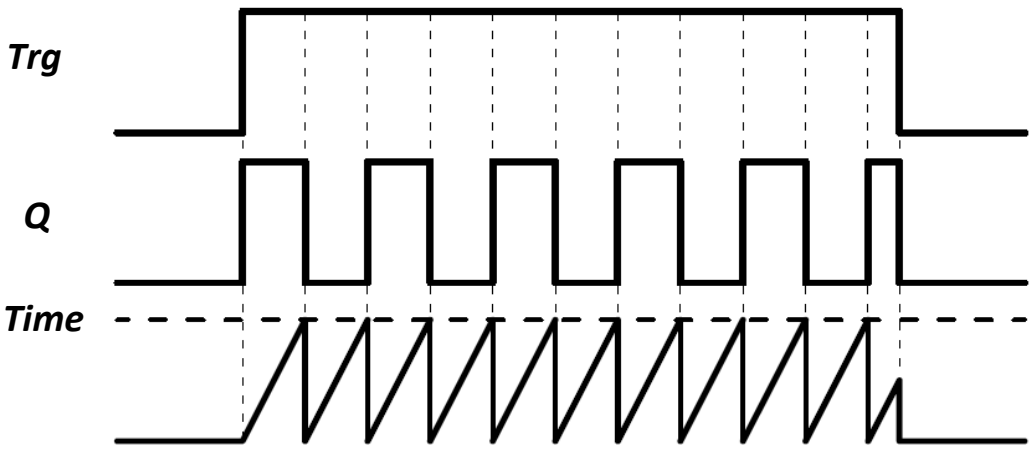


21.2.3 PulseGen Function Block

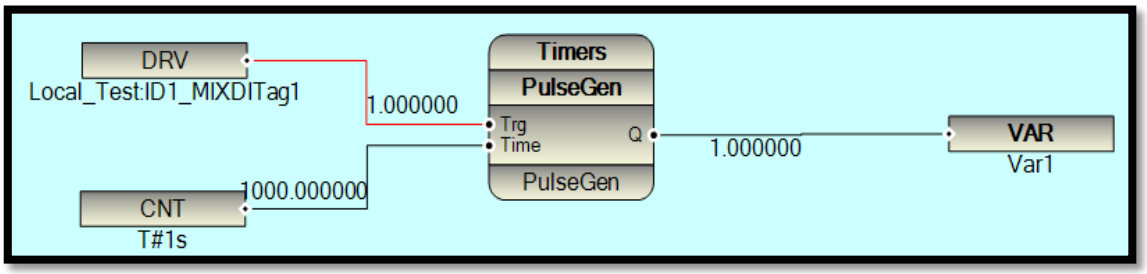
This FB Generate Permanent Pulse in Output Q . When Trg Changed from 0 to 1 , Output Pulse will start .
Q High Time = Time (Msec)
Q Low Time = Time (Msec)
When Trg Changed from 1 to 0 , pulse will stop.

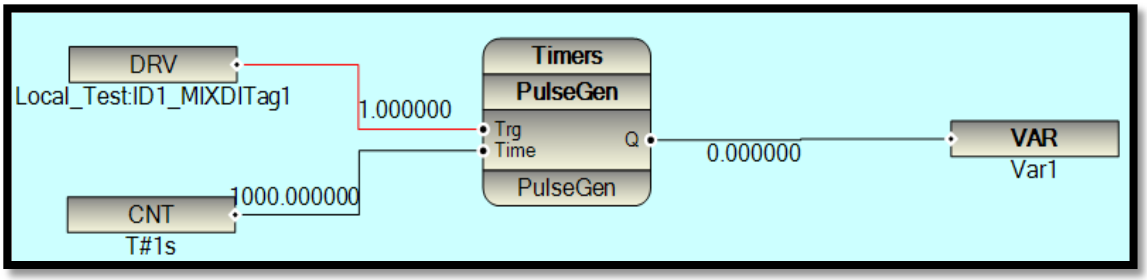
PulseGen			
	**I/O**	**Data Type**	**Description**
	Trg	**Double**	*Timer operation condition*
	Time	**Double**	*Preset Time*
	Q	**Double**	*Timer output*

Timing Diagram



For Example:

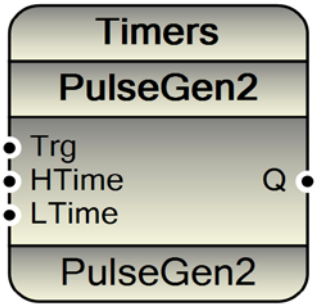




21.2.4 PulseGen2 Function Block

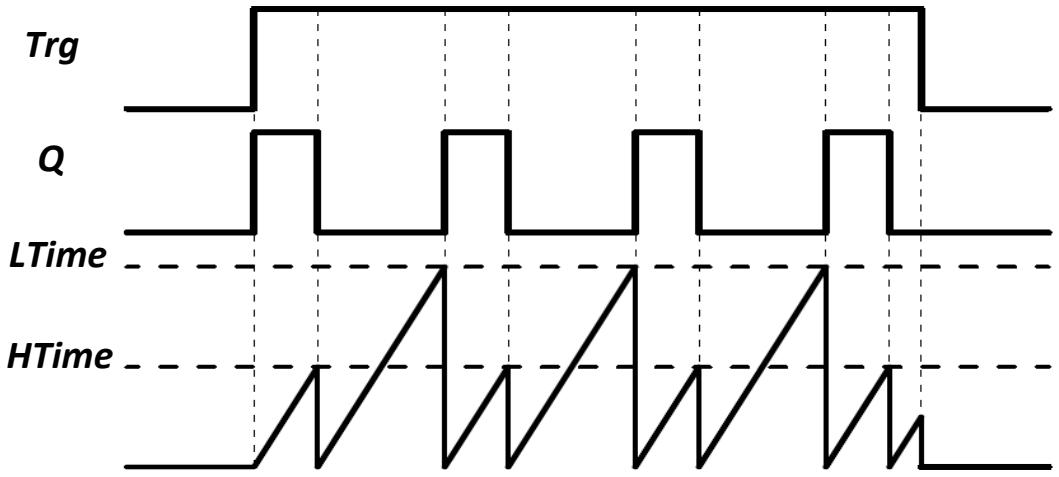
This FB Generate Permanent Pulse in Output Q . When Trg Changed from 0 to 1 , Output Pulse will start .
Q High Time = HTime (Msec)
Q Low Time = LTime (Msec)
When Trg Changed from 1 to 0 , pulse will stop .

PulseGen2

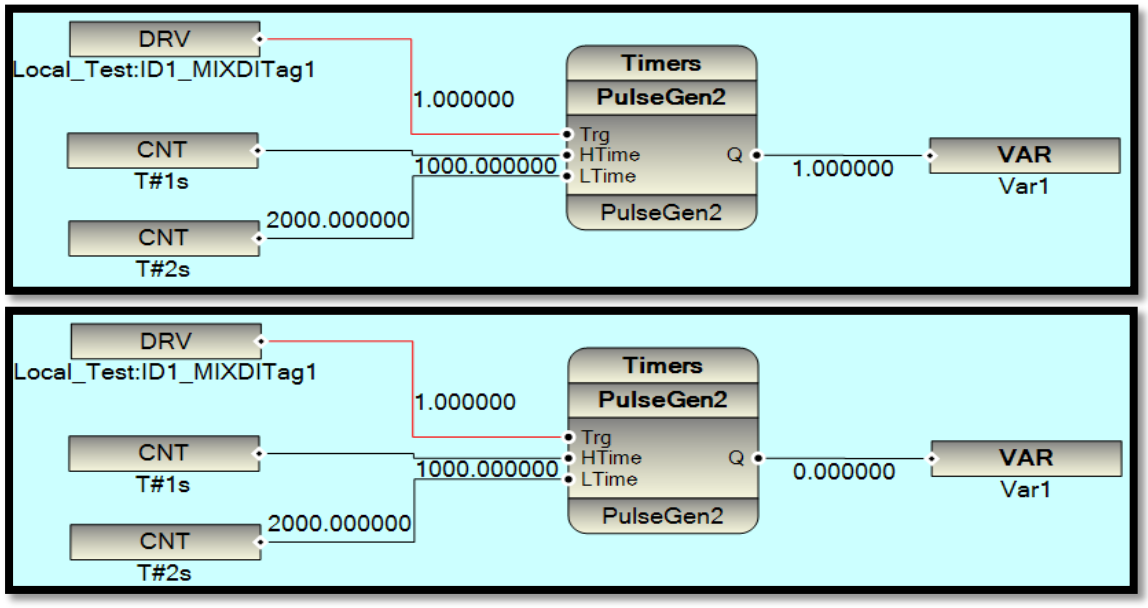


I/O	Data Type	Description
Trg	Double	*Timer operation condition*
HTime	Double	*High Preset Time*
LTime	Double	*Low Preset Time*
Q	Double	*Timer output*

Timing Diagram

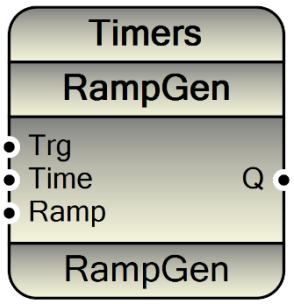


For Example:



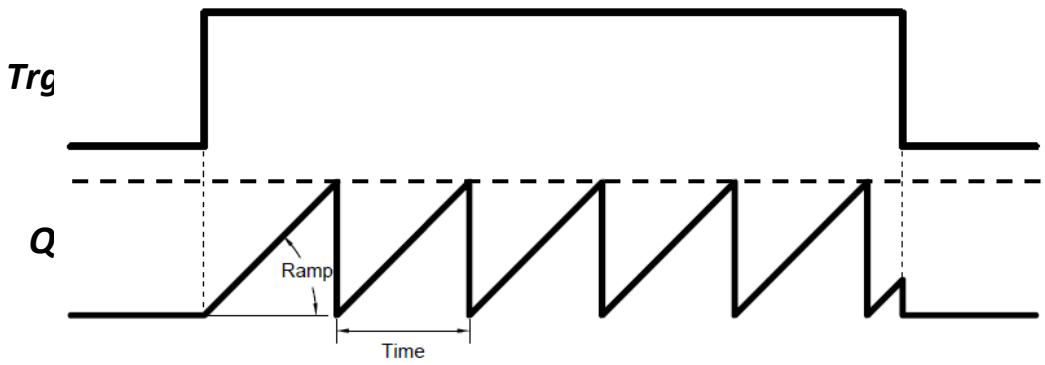
21.2.5 RampGen Function Block

This FB Generate Ramp Output . When Trg Change from 0 to 1, FB Will Start.
Time is in Sec, width of Ramp.
Ramp: Ramp angle in Deg

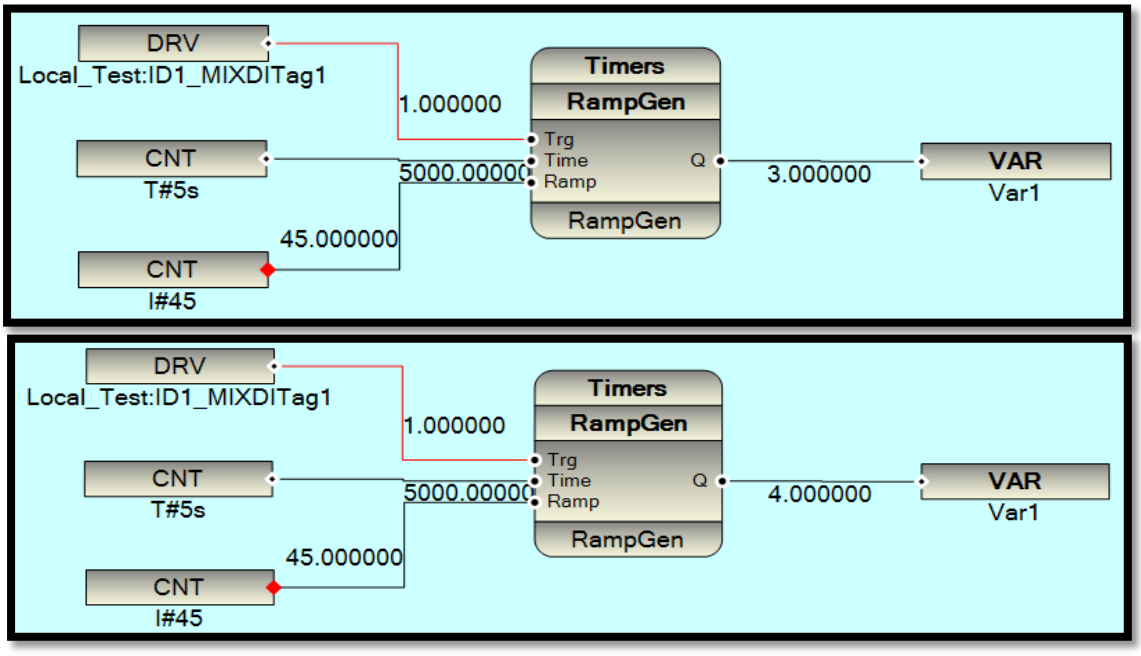


RampGan		
I/O	**Data Type**	**Description**
Trg	**Double**	*Timer operation condition*
Time	**Double**	*Preset Time*
Ramp	**Double**	*Ramp angle in Deg*
Q	**Double**	*Timer output*

Timing Diagram

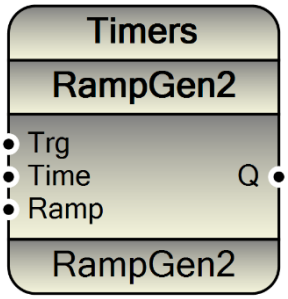
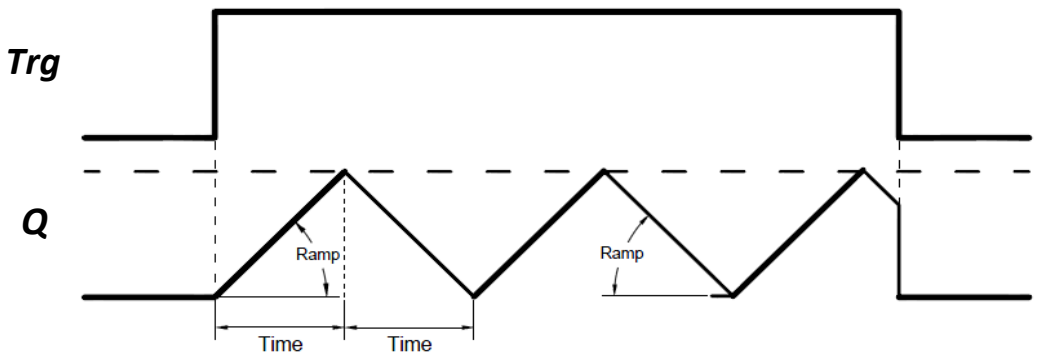


For Example:

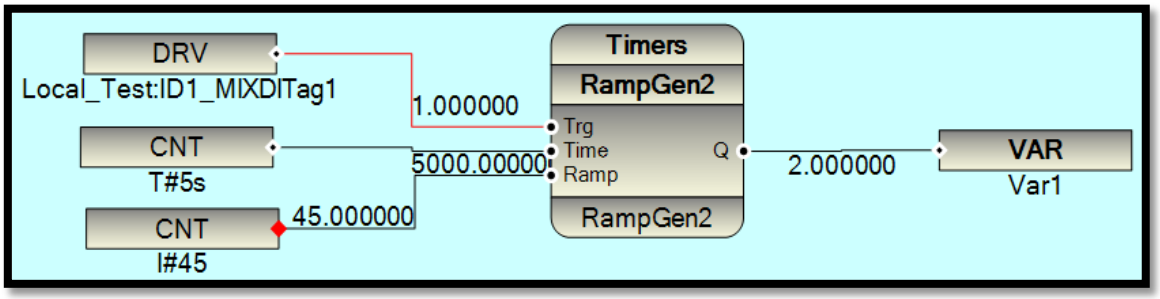


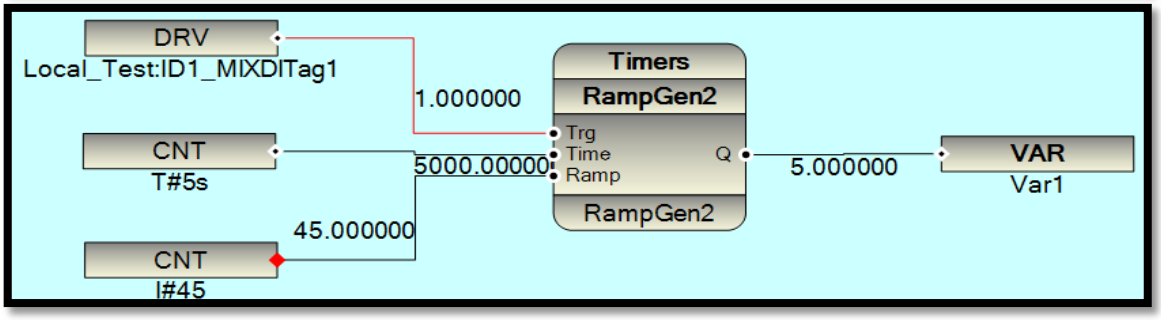
21.2.6 RampGen2 Function Block

This FB Generate Ramp Output .When Trg Change from 0 to 1, FB Will Start.
Time is in Sec, width of Ramp.
Ramp: Ramp angle in Deg

RampGan2			
	**I/O**	**Data Type**	**Description**
	Trg	**Double**	*Timer operation condition*
	Time	**Double**	*Preset Time*
	Ramp	**Double**	*Ramp angle in Deg*
	Q	**Double**	*Timer output*
Timing Diagram			
			

For Example:





21.2.7 SinGen Function Block

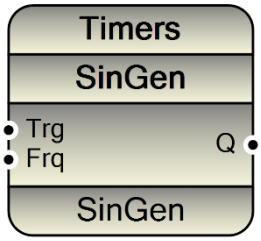
This FB Generate Sine Output

Trg : When change from 0 to 1 , FB start to make Sine output

Frq : Sine Frequency

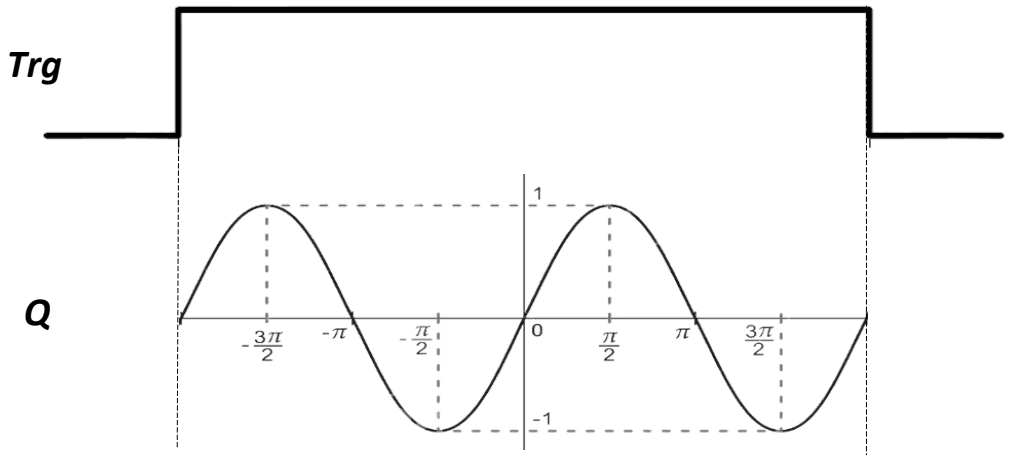
Q : Sine Signal

SinGen

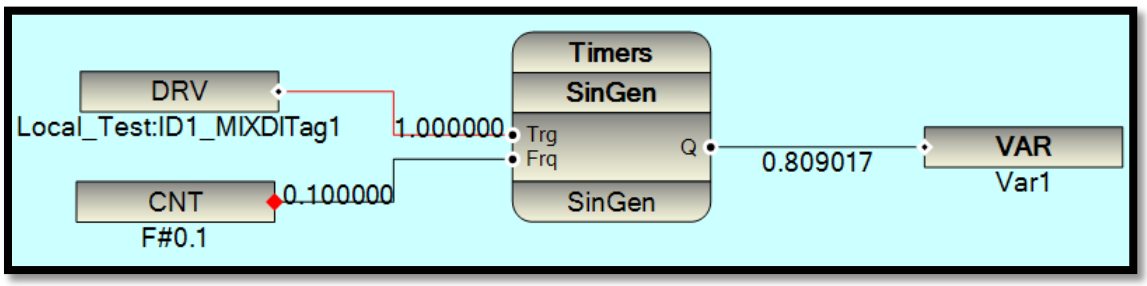


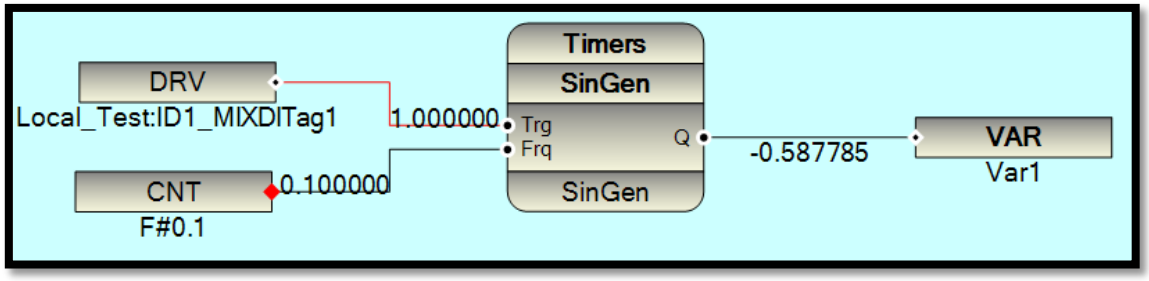
I/O	Data Type	Description
Trg	Double	*Timer operation condition*
Frq	Double	*Sine Frequency*
Q	Double	*Sine Signal*

Timing Diagram



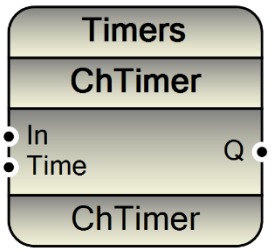
For Example:



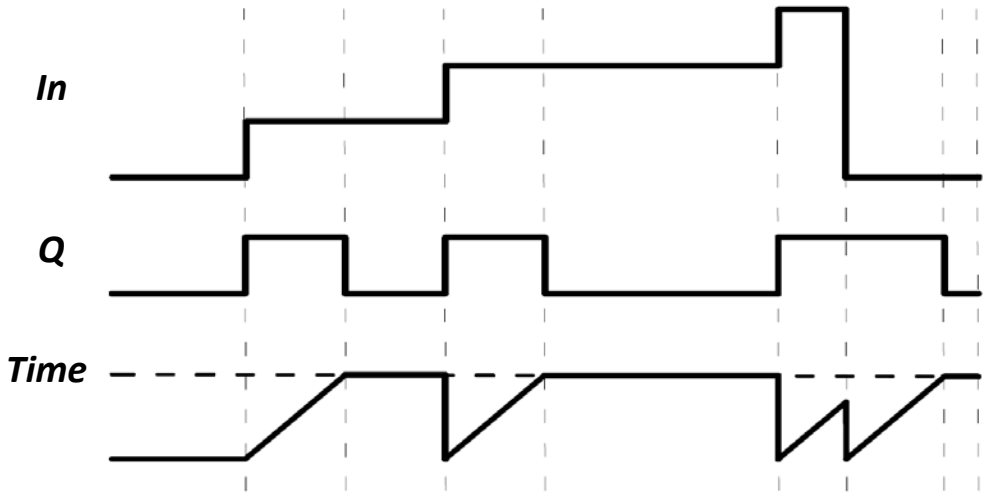


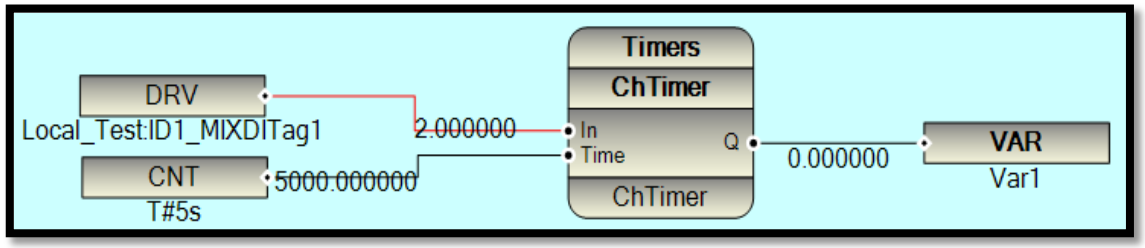
21.2.8 ChTimer Function Block

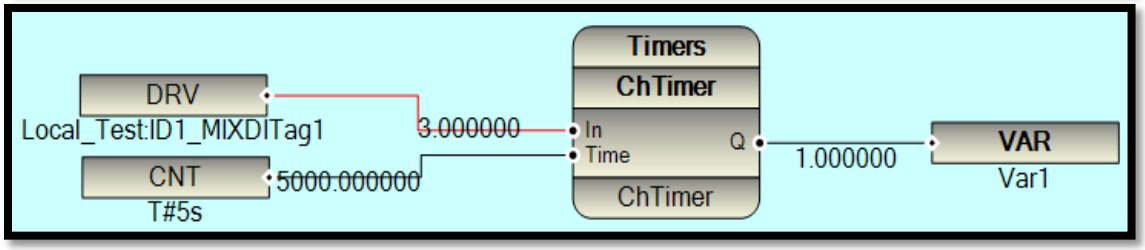
When In Input changing , Q Output will set to 1 for Time Value . in following sample when In signal is changing from 0 to 3 , Q is set for 5 sec and Q will fall after 5 sec .

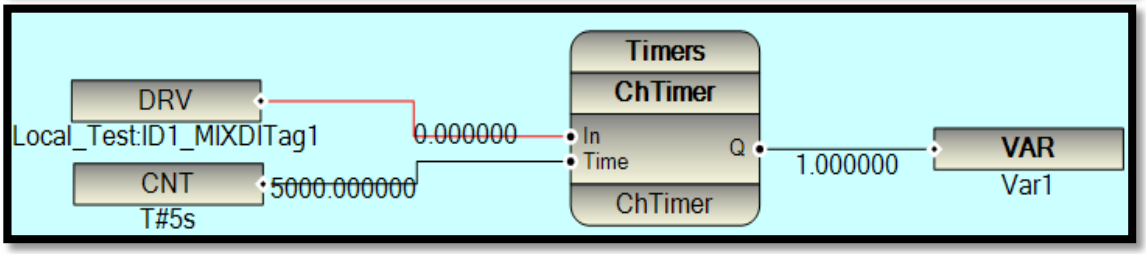
ChTimer			
	**I/O**	**Data Type**	**Description**
	In	**Double**	*Timer operation condition*
	Time	**Double**	*Preset Time*
	Q	**Double**	*Timer output*

Timing Diagram





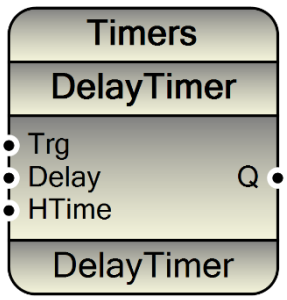




21.2.9 DelayTimer Function Block

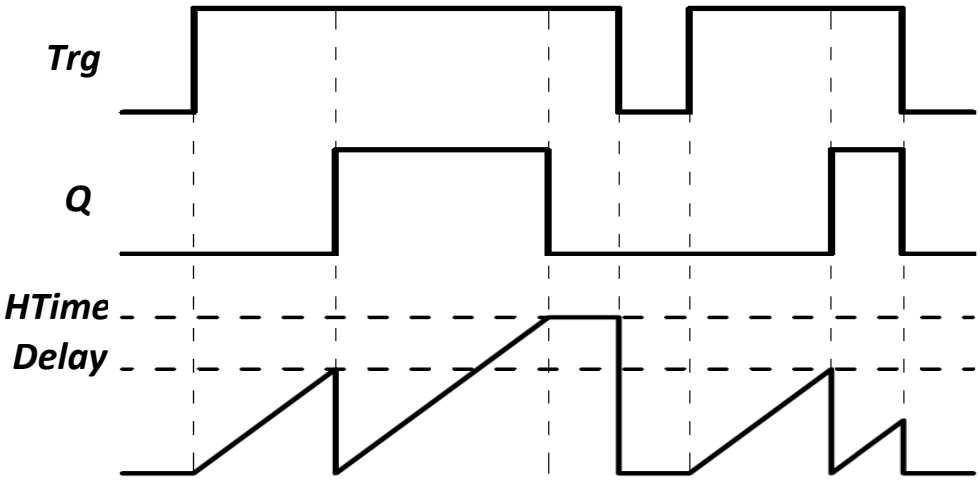
When Trg Input change from 0 to 1 , after Delay Time , Q will change to 1 for HTime .

DelayTimer

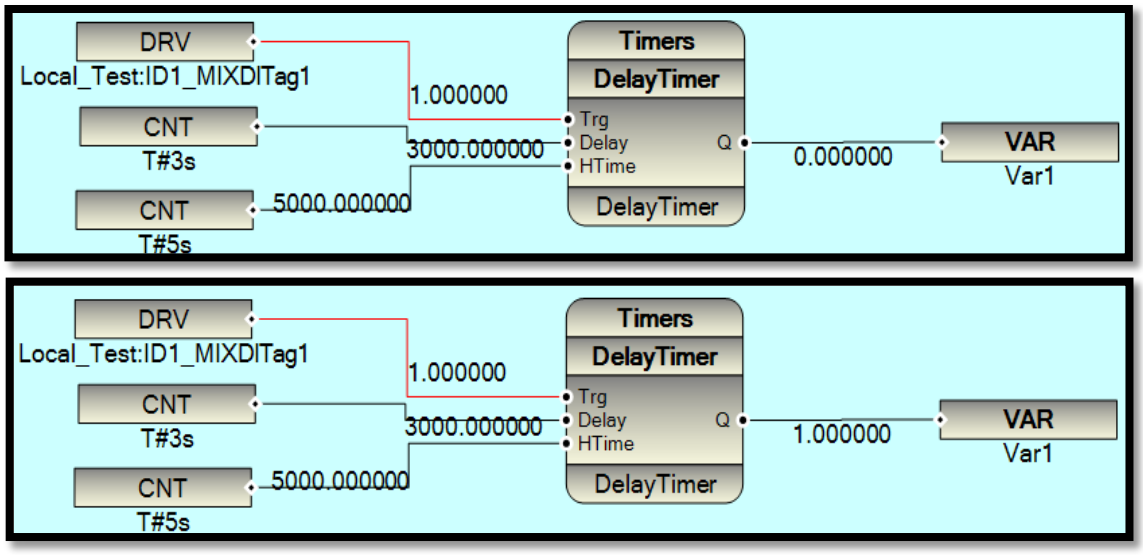


I/O	Data Type	Description
Trg	Double	*Timer operation condition*
Delay	Double	*Low Preset Time*
HTime	Double	*High Preset Time*
Q	Double	*Timer output*

Timing Diagram

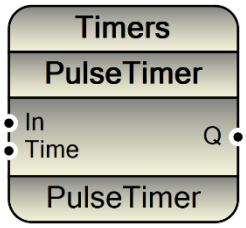
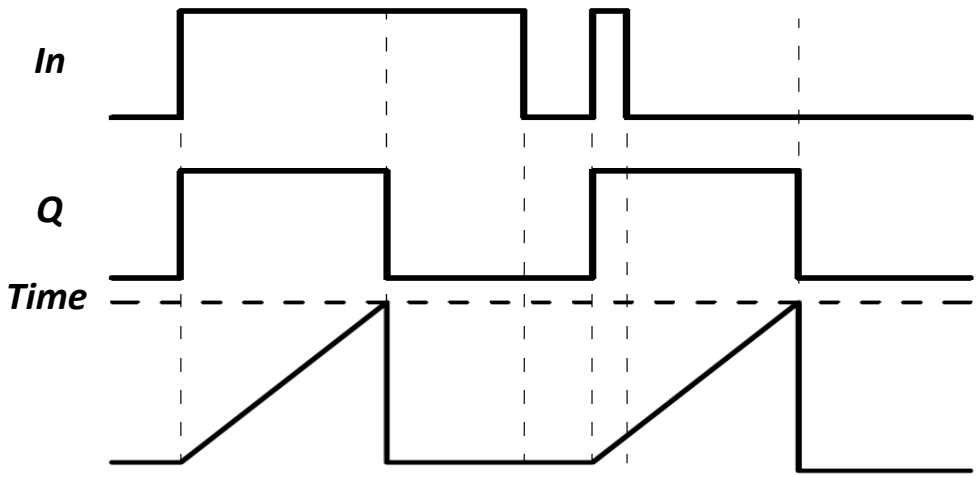


For Example:

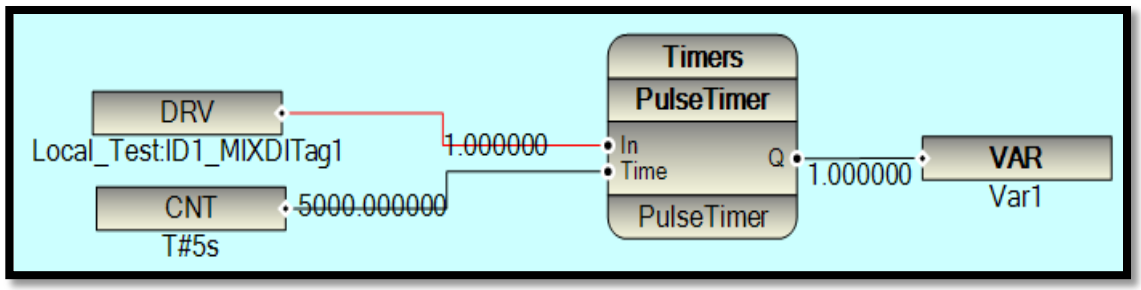


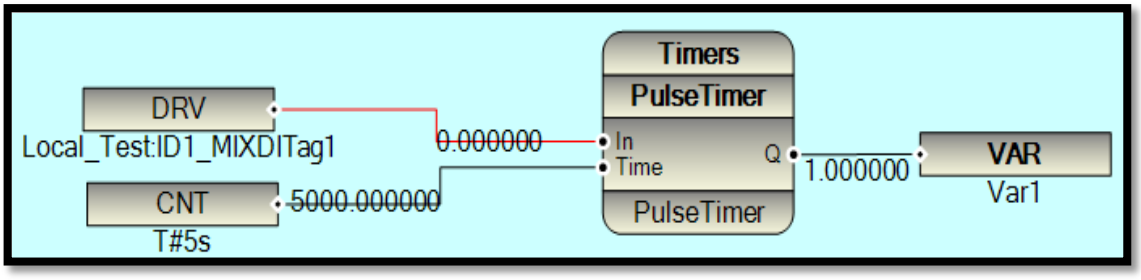
21.2.10 PulseTimer Function Block

When Trg Input change from 0 to 1 ,Q will change to 1 for Time .In PulseTimer ,output is independed of input and just trige with rising edge.

PulseTimer			
	**I/O**	**Data Type**	**Description**
	In	**Double**	*Timer operation condition*
	Time	**Double**	*Preset Time*
	Q	**Double**	*Timer output*
Timing Diagram			
			

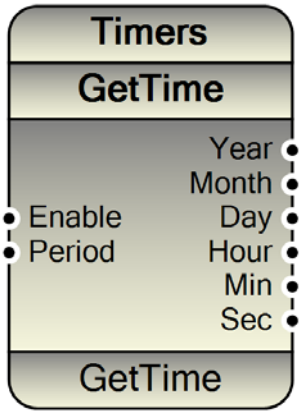
For Example:



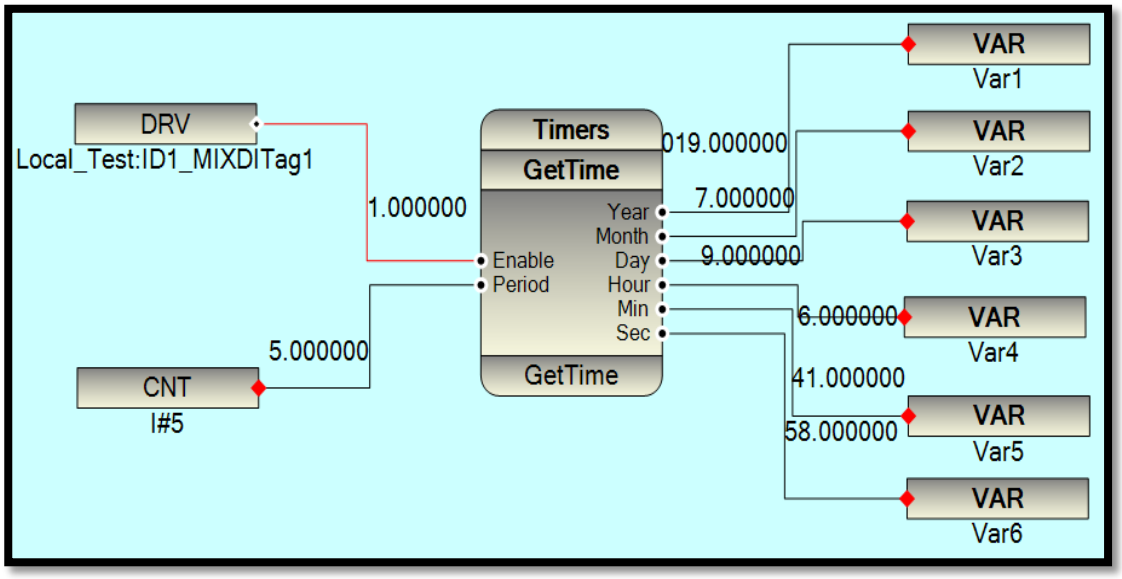


21.2.11 GetTime Function Block

When Trg Input change from 0 to 1 ,GetTime FB show current time.also by set time in period can refresh time periodically.

GetTime			
	**I/O**	**Data Type**	**Description**
	Enable	**Double**	*Timer operation condition*
	Period	**Double**	*Refresh Time*
	Year	**Double**	*Year*
	Month	**Double**	*Month*
	Day	**Double**	*Day*
	Hour	**Double**	*Hour*
	Min	**Double**	*Minuets*
	Sec	**Double**	*Second*

For Example:



21.3 Counter Group:

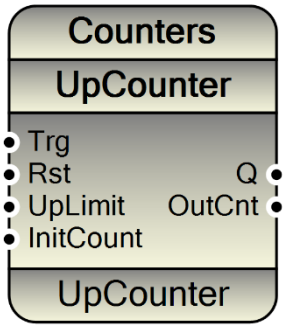
Counter Function Groups

21.3.1 UpCounter Function Block

When Trg Input Change from 0 to 1 , OutCnt will increase by One . When OutCnt reach to UpLimit then Q will set to 1 and OutCnt will not change any more , until Rst Signal is changed from 0 to 1 .

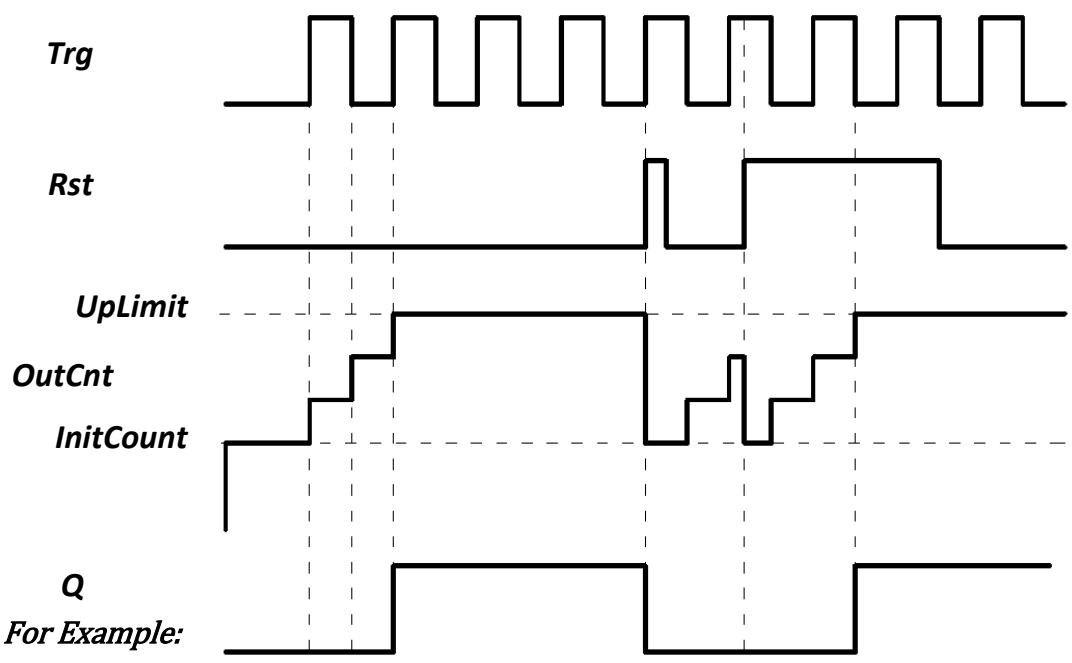
-When Trg Changed from 0 to 1 : If OutCnt is not reached to UpLimit , OutCnt increase by one
- When Rst is changed from 0 to1: OutCnt set to InitCount , Q set to 0
- If OutCnt reach to UpLimit then Q will set to 1

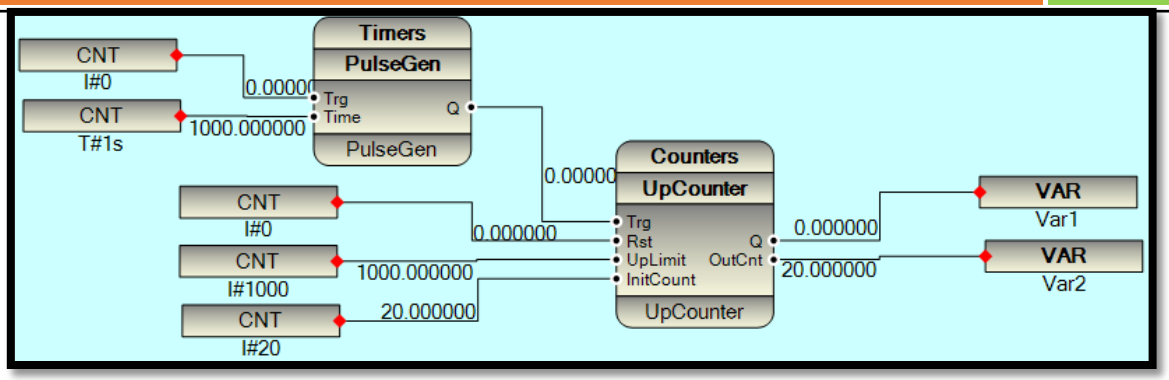
UpCounter

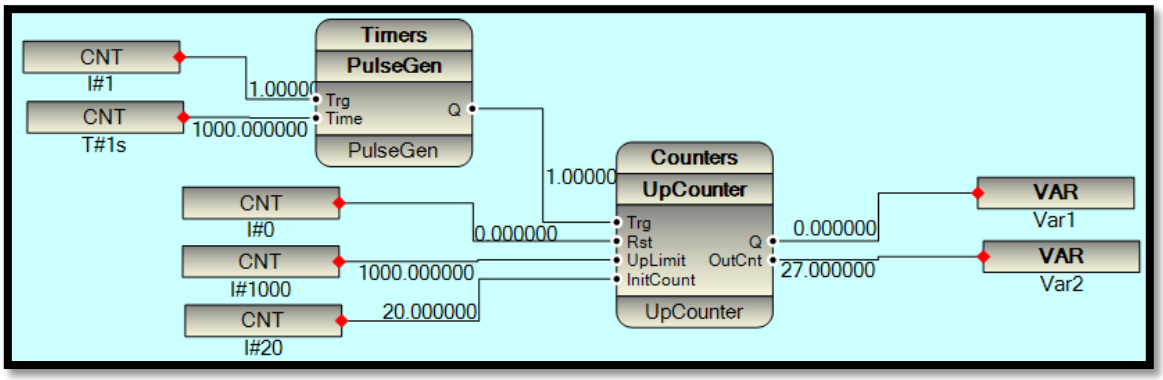


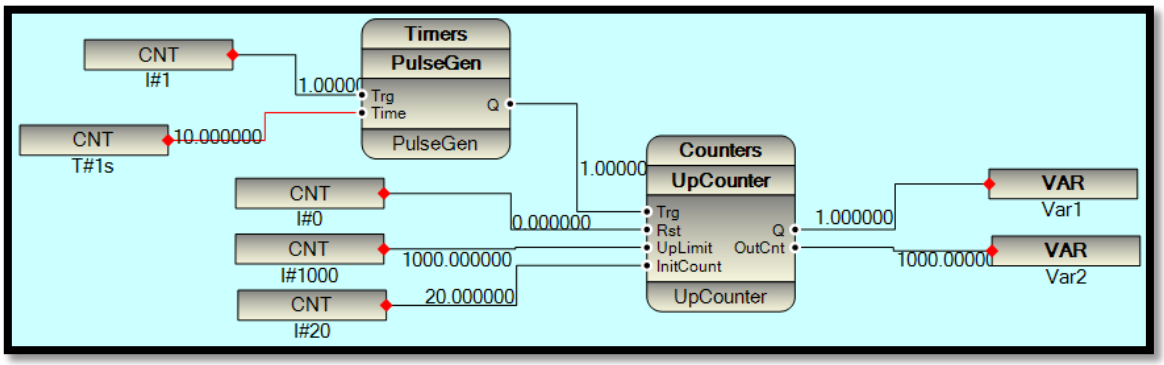
I/O	Data Type	Description
Trg	Double	*Counter Input*
Rst	Double	*Reset*
UpLimit	Double	*maximum quantity of counter*
InitCount	Double	*Minimum quantity of counter*
Q	Double	*Counter Output*
QutCut	Double	*quantity of counter*

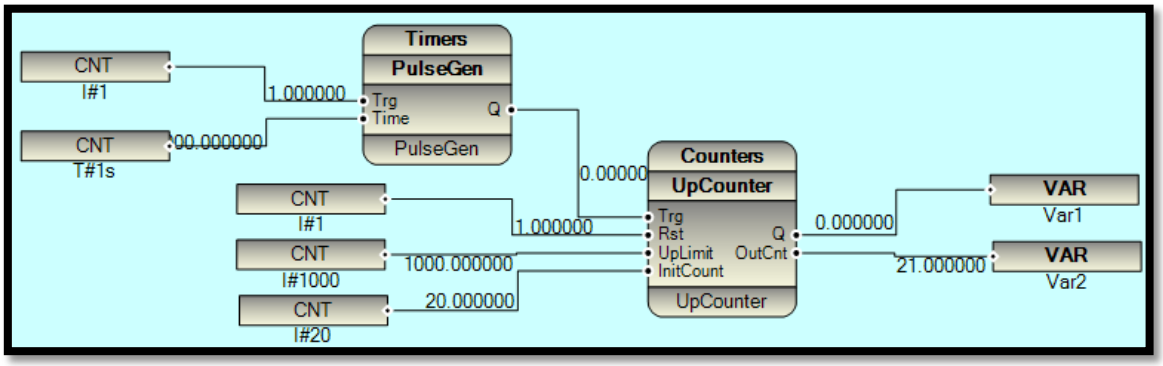
Counting Diagram




CNT
I#0
0.00000
CNT
T#1s
1000.000000
Timers
PulseGen
Trg
Time
Q
PulseGen
0.00000
Counters
UpCounter
Trg
Rst
UpLimit
InitCount
Q
OutCnt
UpCounter
CNT
I#0
0.000000
CNT
I#1000
1000.000000
CNT
I#20
20.000000
0.000000
20.000000
VAR
Var1
VAR
Var2


CNT
I#1
1.00000
CNT
T#1s
1000.000000
Timers
PulseGen
Trg
Time
Q
PulseGen
1.00000
Counters
UpCounter
Trg
Rst
UpLimit
InitCount
Q
OutCnt
UpCounter
CNT
I#0
0.000000
CNT
I#1000
1000.000000
CNT
I#20
20.000000
0.000000
27.000000
VAR
Var1
VAR
Var2


CNT
I#1
1.00000
CNT
T#1s
10.000000
Timers
PulseGen
Trg
Time
Q
PulseGen
1.00000
Counters
UpCounter
Trg
Rst
UpLimit
InitCount
Q
OutCnt
UpCounter
CNT
I#0
0.000000
CNT
I#1000
1000.000000
CNT
I#20
20.000000
1.000000
1000.00000
VAR
Var1
VAR
Var2

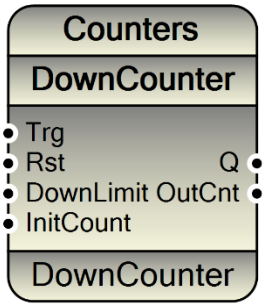

CNT
I#1
1.000000
CNT
T#1s
00.000000
Timers
PulseGen
Trg
Time
Q
PulseGen
0.00000
Counters
UpCounter
Trg
Rst
UpLimit
InitCount
Q
OutCnt
UpCounter
CNT
I#1
1.000000
CNT
I#1000
1000.000000
CNT
I#20
20.000000
0.000000
21.000000
VAR
Var1
VAR
Var2

21.3.2 DownCounter Function Block

When Function block is run for first time (There is no Static data for FB) , OutCnt will set to InitCount .When Trg changed from 0 to 1 , OutCnt will decrease by one . When OutCnt reached to DownLimit , then Q will set to 1 and OutCnt will not changed until Rst Input changed from 0 to 1 .

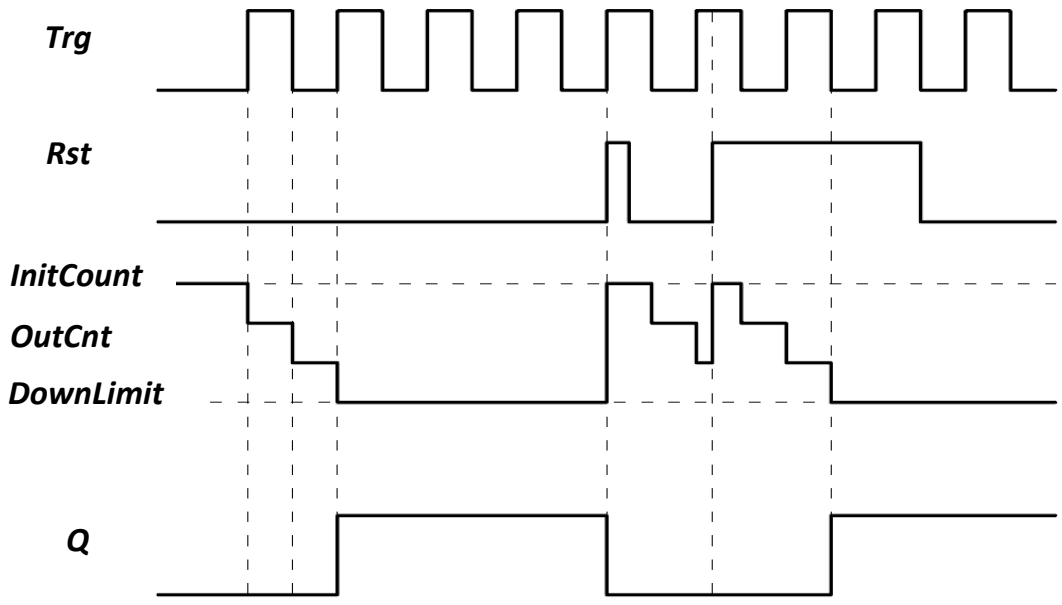
-When Rst Input changed from 0 to 1: OutCnt will set to InitCount ,Q will set to 0
-When Trg Changed from 0 to 1 , If OutCnt is bigger than DownLimit , Outcnt will decrease by one , Q is set to 0
-When Trg Changed from 0 to 1 , If OutCnt is equal to DownLimit , Outcnt will set to DownLimit , Q will set to 1

DownCounter

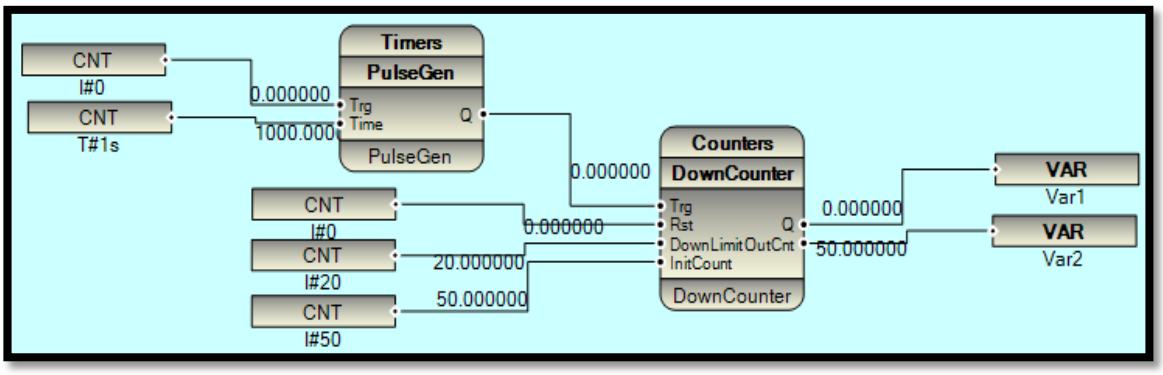


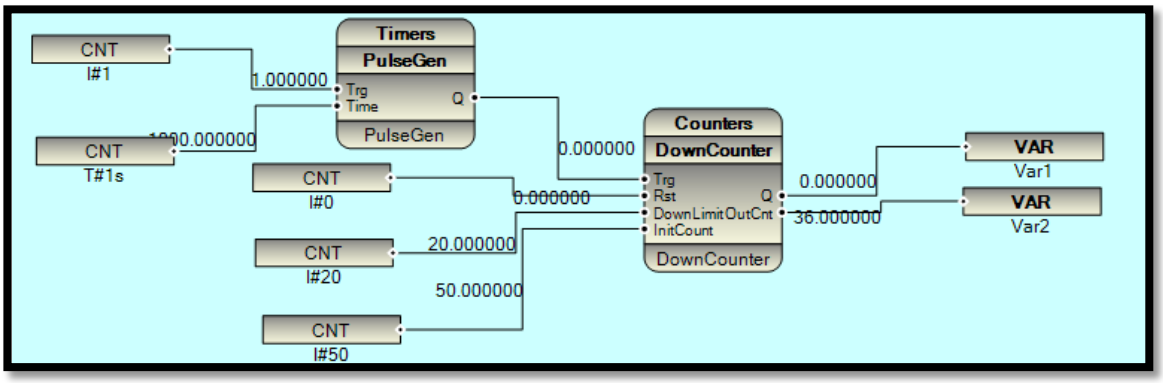
I/O	Data Type	Description
Trg	Double	*Counter Input*
Rst	Double	*Reset*
DownLimit	Double	*maximum quantity of counter*
InitCount	Double	*Minimum quantity of counter*
Q	Double	*Counter Output*
QutCut	Double	*quantity of counter*

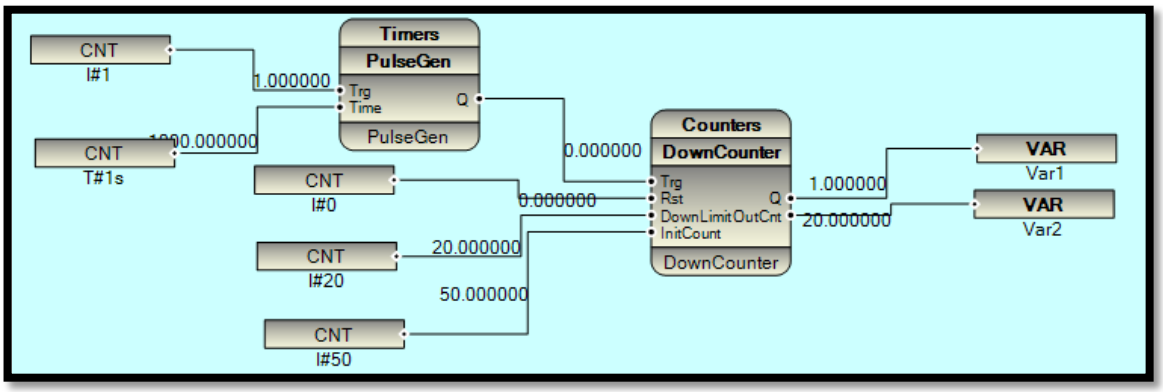
Counting Diagram

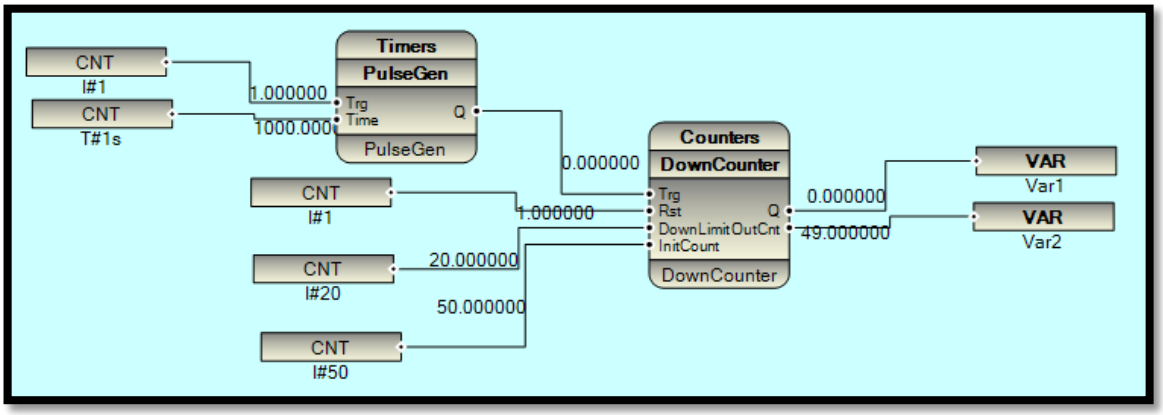


For Example:


CNT
I#0
CNT
T#1s
0.000000
1000.000
Timers
PulseGen
Trg
Time
Q
PulseGen
0.000000
Counters
DownCounter
Trg
Rst
Q
DownLimitOutCnt
InitCount
DownCounter
0.000000
50.000000
VAR
Var1
VAR
Var2
CNT
I#0
CNT
I#20
20.000000
CNT
I#50
50.000000


CNT
I#1
1.000000
CNT
T#1s
Timers
PulseGen
Trg
Time
Q
PulseGen
0.000000
Counters
DownCounter
Trg
Rst
Q
DownLimitOutCnt
InitCount
DownCounter
0.000000
36.000000
VAR
Var1
VAR
Var2
CNT
I#0
0.000000
CNT
I#20
20.000000
50.000000
CNT
I#50


CNT
I#1
1.000000
CNT
T#1s
Timers
PulseGen
Trg
Time
Q
PulseGen
0.000000
Counters
DownCounter
Trg
Rst
Q
DownLimitOutCnt
InitCount
DownCounter
1.000000
20.000000
VAR
Var1
VAR
Var2
CNT
I#0
0.000000
CNT
I#20
20.000000
50.000000
CNT
I#50

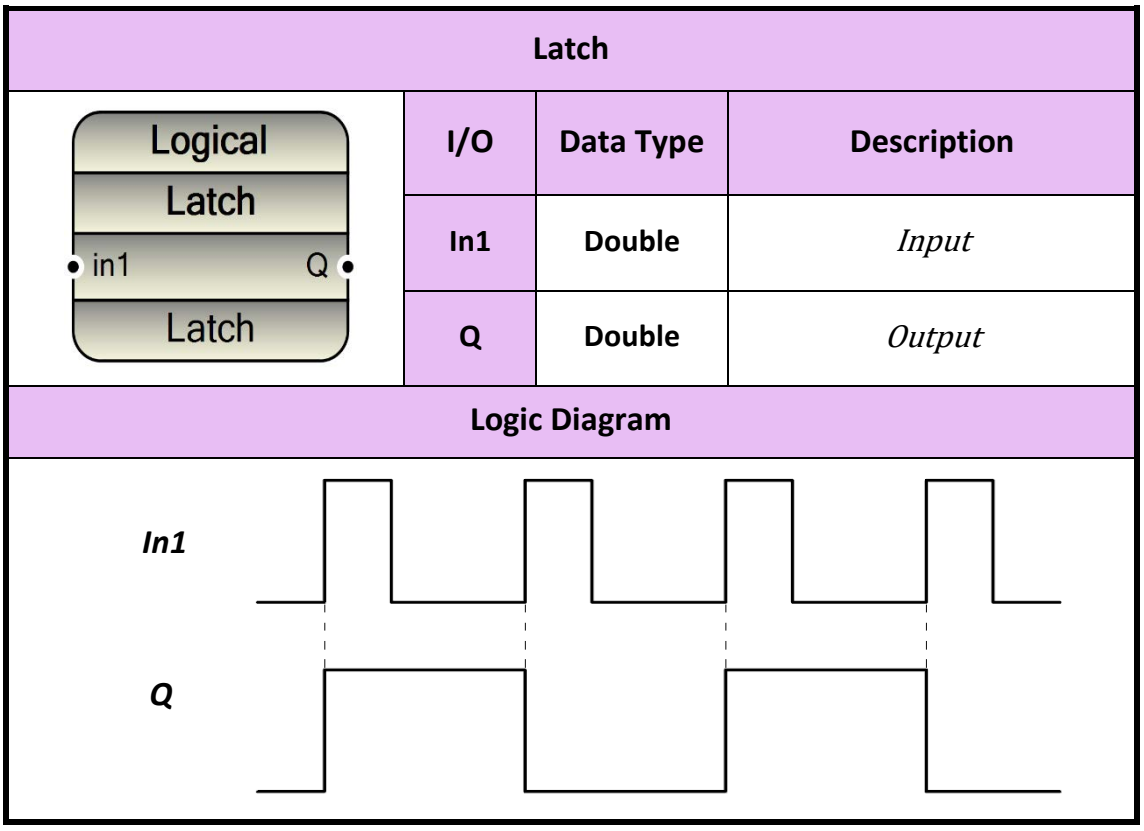

CNT
I#1
CNT
T#1s
1.000000
1000.000
Timers
PulseGen
Trg
Time
Q
PulseGen
0.000000
Counters
DownCounter
Trg
Rst
Q
DownLimitOutCnt
InitCount
DownCounter
0.000000
49.000000
VAR
Var1
VAR
Var2
CNT
I#1
1.000000
CNT
I#20
20.000000
50.000000
CNT
I#50

21.4 Logic Group:

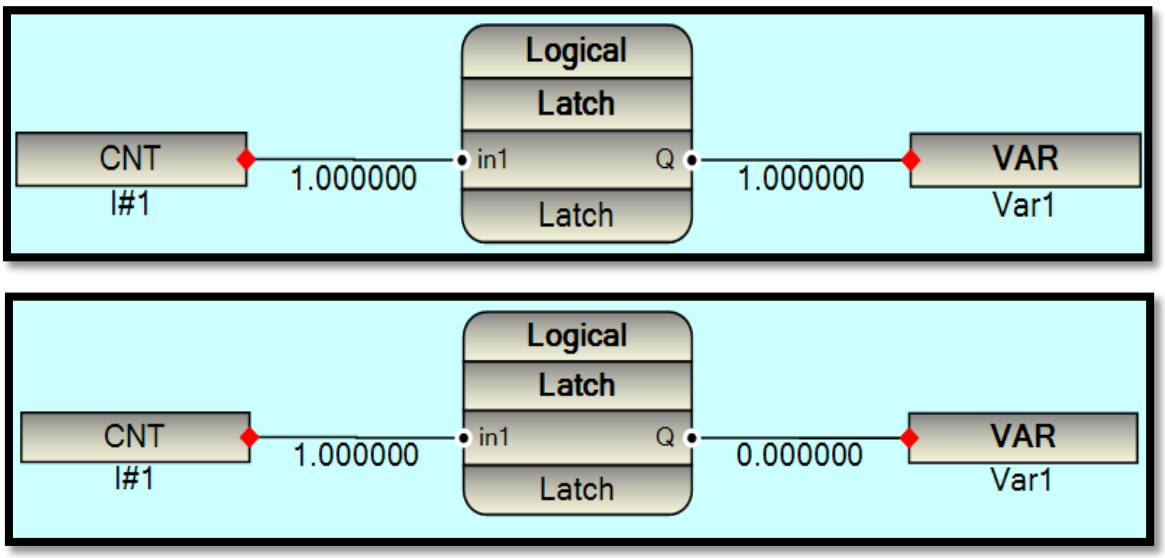
Logic Function Groups

21.4.1 Latch Function Block

When In1 changed from 0 to 1, Q will set to 1. Q is set to 1 until In1 is changed again from 0 to 1.



For Example:



21.4.2 RS_FF Function Block

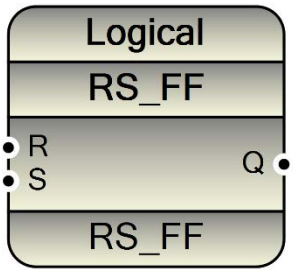
Reset Set Flip Flop.

When R Input changed from 0 to 1: Q will set to 0

When S Input Changed from 0 to 1: Q will set to 1

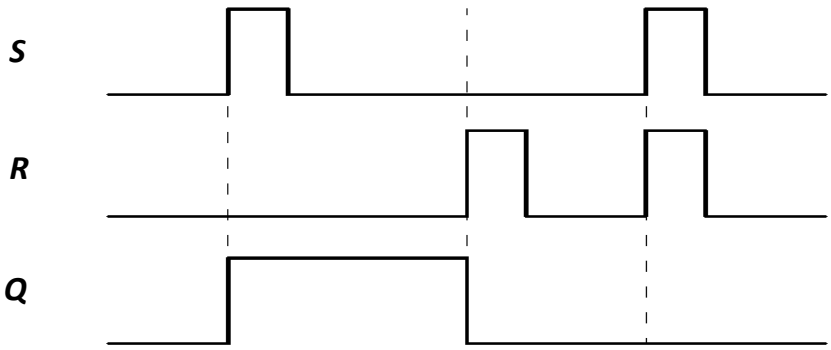
If R and S changed from 0 to 1 at same time, Q will set to 0.

RS_FF

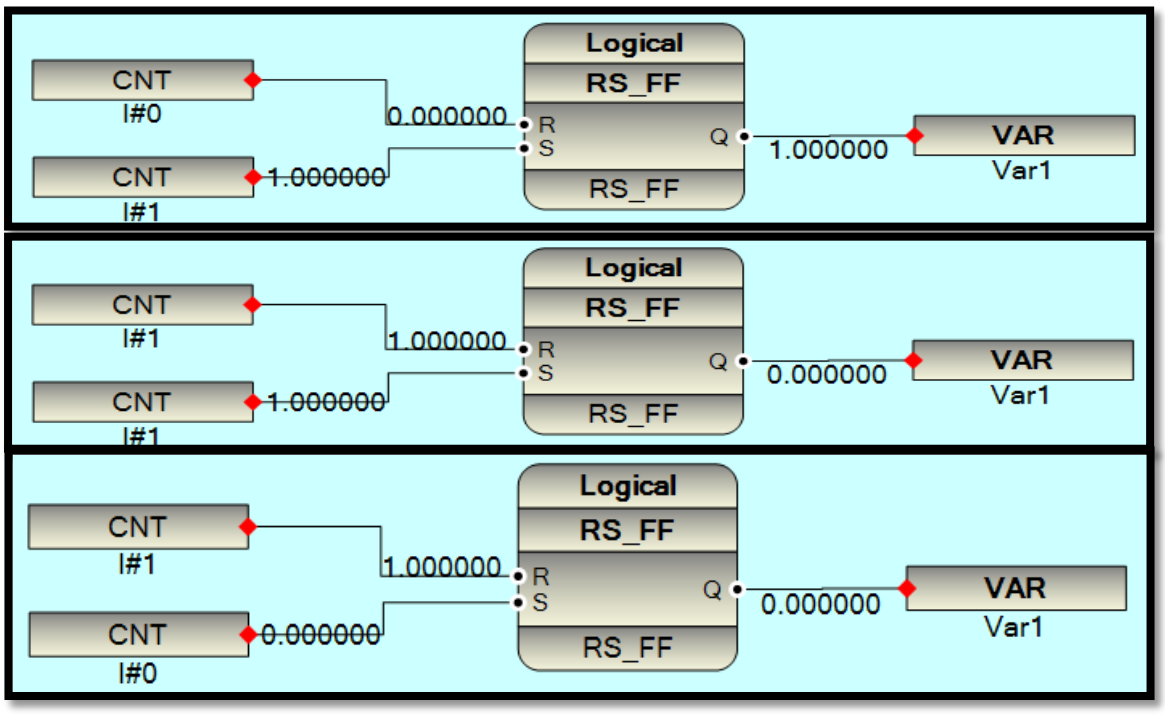


I/O	Data Type	Description
R	Double	*Reset*
S	Double	*Set*
Q	Double	*Output*

Logic Diagram



For Example:



21.4.3 SR_FF Function Block

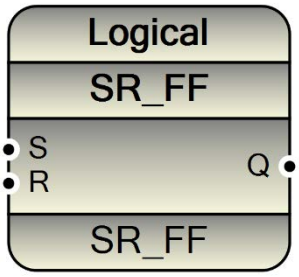
Set Reset Flip Flop.

When S Input Changed from 0 to 1: Q will set to 1

When R Input changed from 0 to 1: Q will set to 0

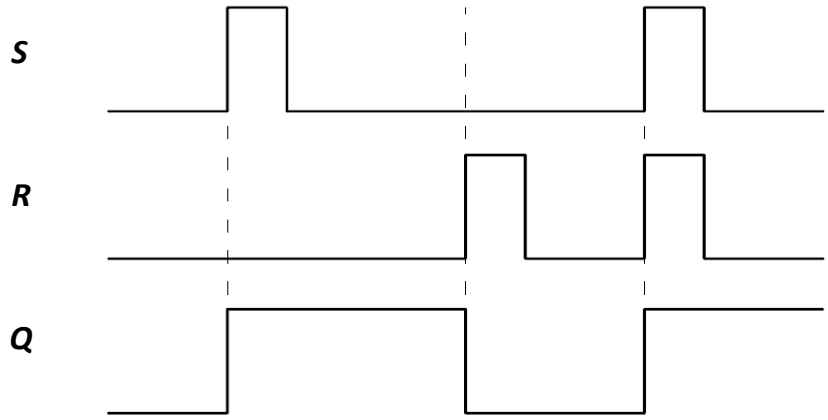
If R and S changed from 0 to 1 at same time, Q will set to 1.

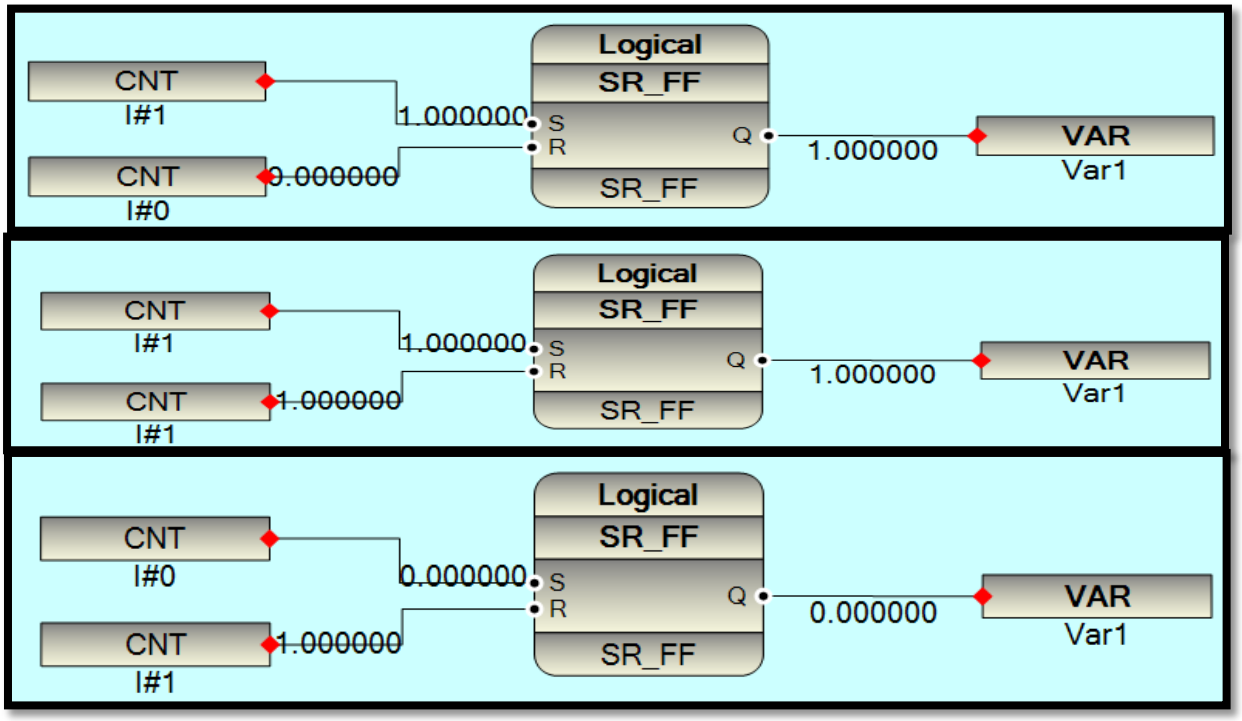
SR_FF



I/O	Data Type	Description
S	Double	*Set*
R	Double	*Reset*
Q	Double	*Output*

Logic Diagram





21.4.4 JK_FF Function Block

JK Flip Flop.

If J is changed from 0 to 1: Q will set to 1

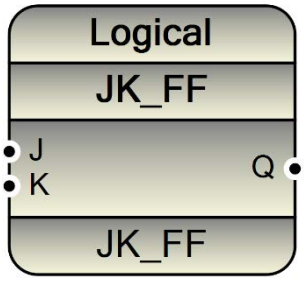
If K is changed from 0 to 1: Q will set to 0

If J & K changed for m0 to 1 In the same time:

- If Q is 1, Q will set to 0.

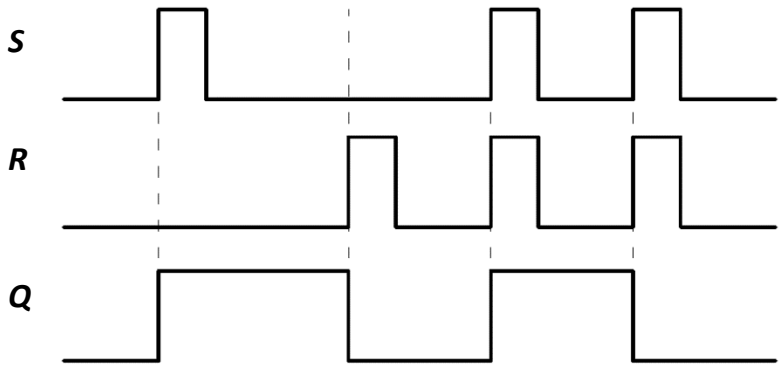
- If Q is 0, Q will set to 1.

JK_FF

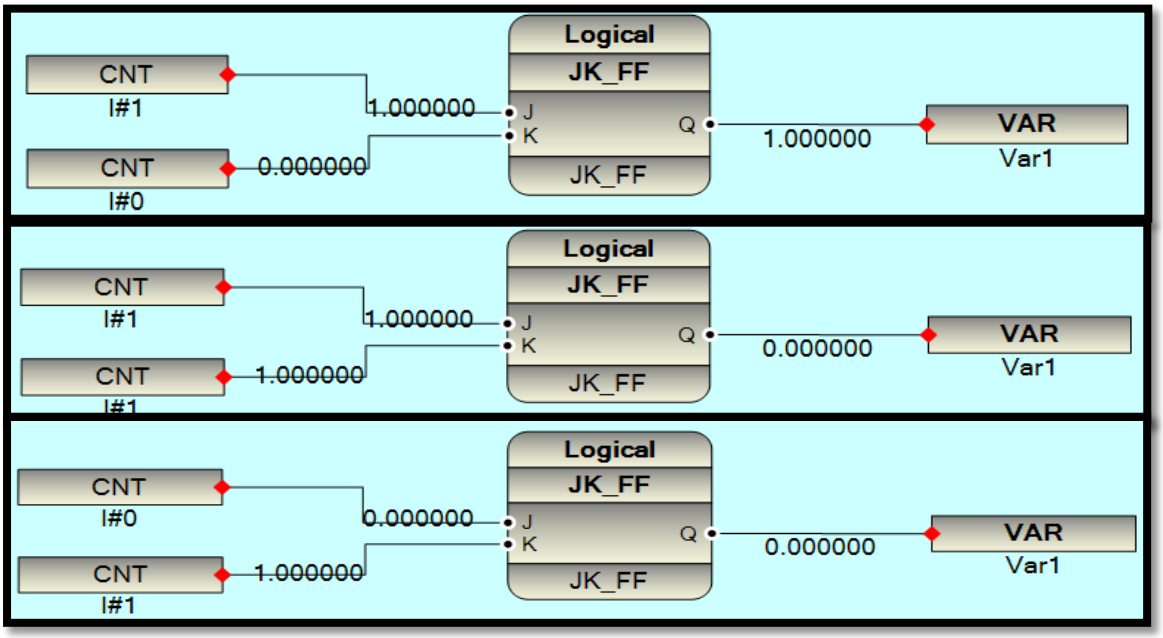


I/O	Data Type	Description
J	Double	*Set*
K	Double	*Reset*
Q	Double	*Output*

Logic Diagram

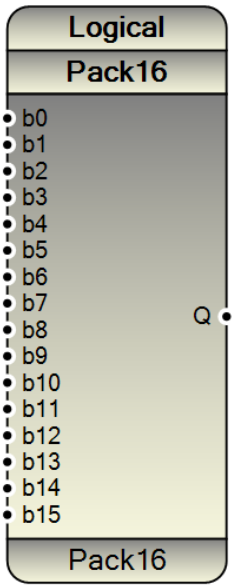


For Example:

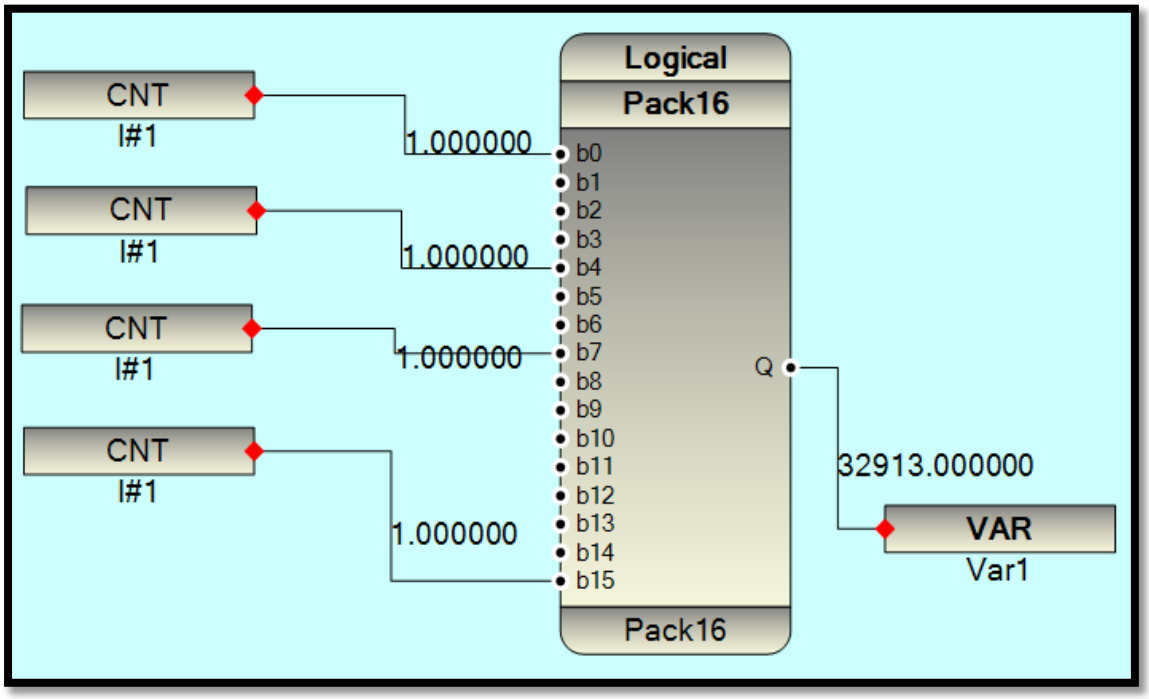


21.4.5 Pack16 Function Block

Pack16 will combine 16 digital signal (0 or 1) to one 16 Bit integer value.

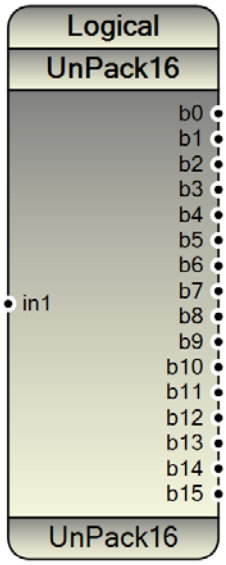
Pack16			
	**I/O**	**Data Type**	**Description**
	b0,…, b15	**Double**	*Digital Input*
	Q	**Double**	*Output(Integer value)*

For Example: $(1000000010010001)_2 = 32913$

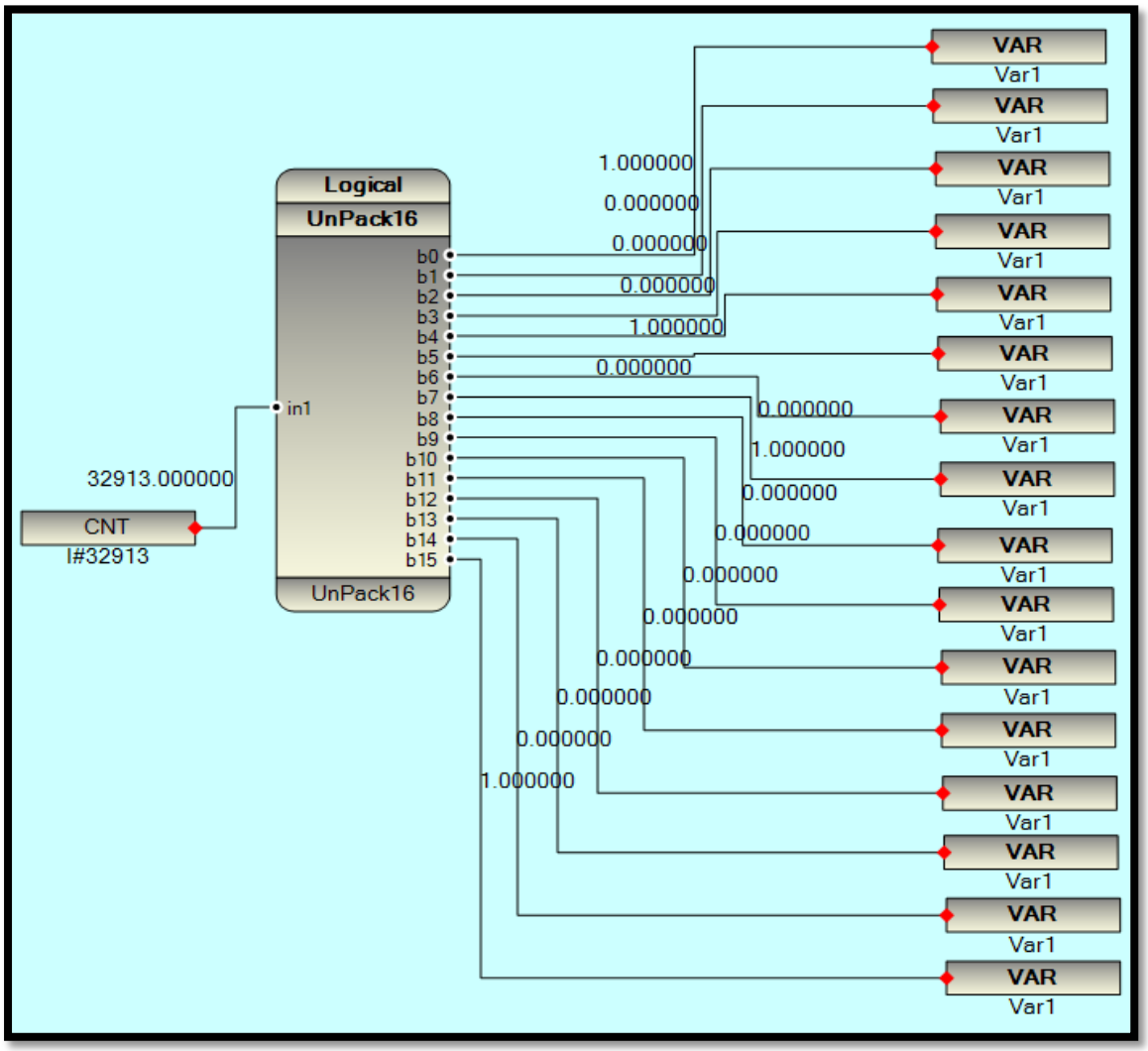


21.4.6 UnPack16 Function Block

UnPack16 will Convert one 16-bit integer to 16 digital signal (0 or 1)

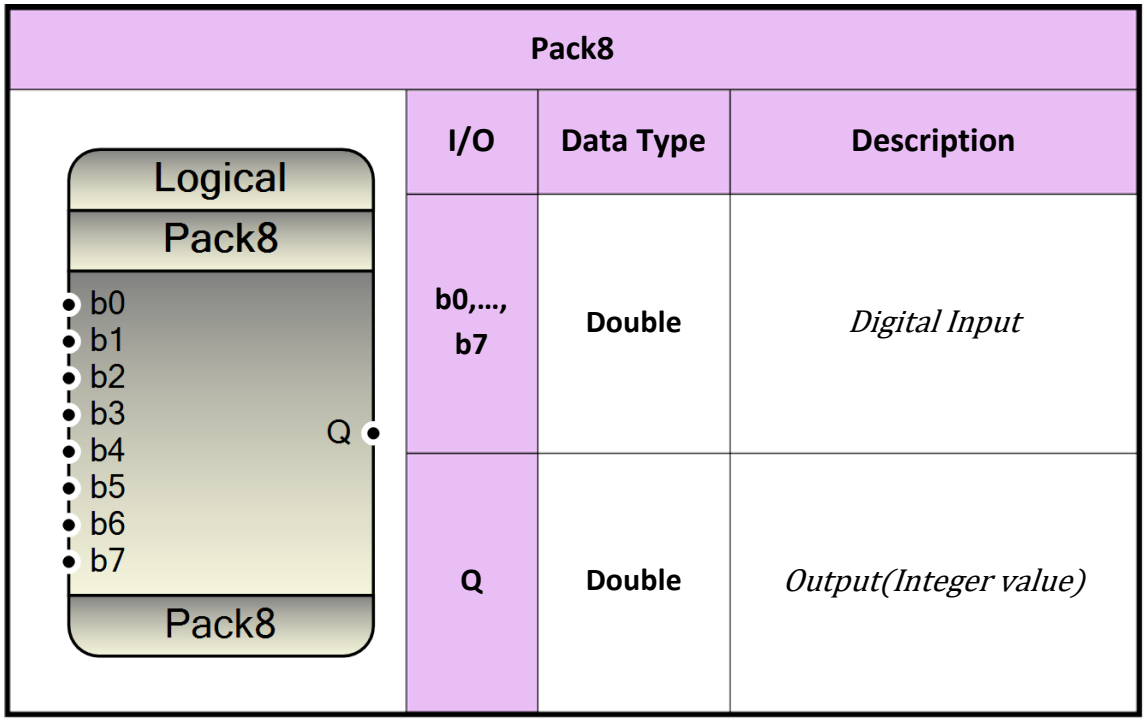
UnPack16			
	**I/O**	**Data Type**	**Description**
	In1	**Double**	*Input(Integer value)*
	b0,…, b15	**Double**	*Digital Output*

For Example: $32913 = (1000000010010001)_2$



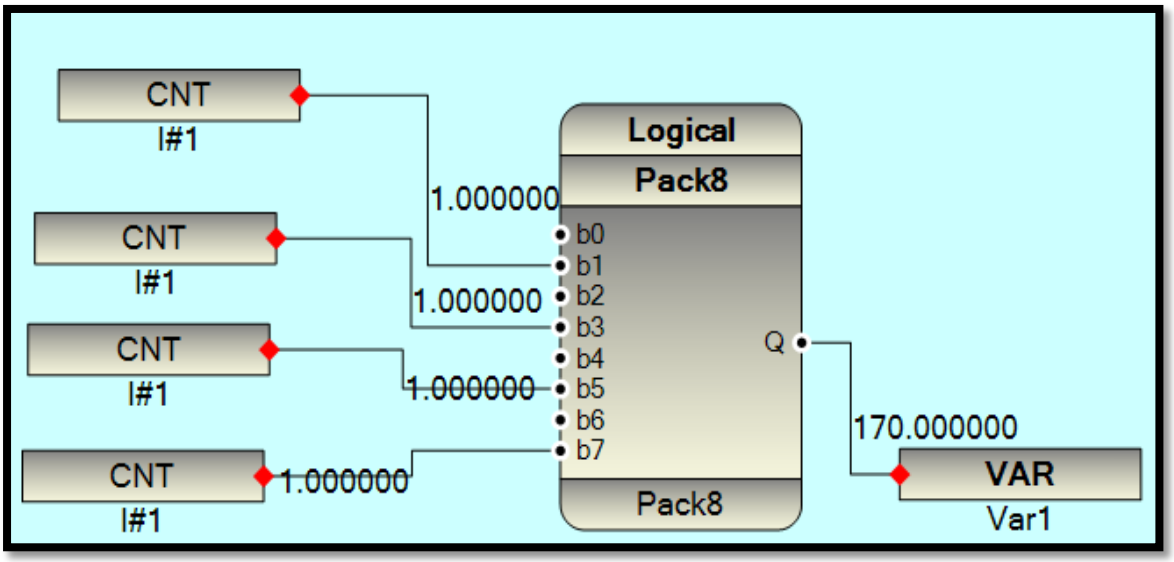
21.4.7 Pack8 Function Block

Pack8 will combine 8 digital signal (0 or 1) to one 8 Bit integer value.



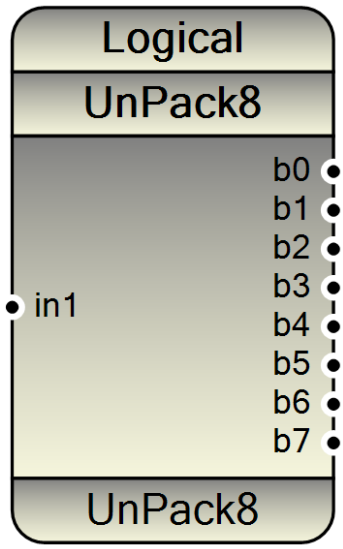
Pack8		
I/O	**Data Type**	**Description**
b0,…, b7	**Double**	*Digital Input*
Q	**Double**	*Output(Integer value)*

For Example: $(101001010)_2 = 170$

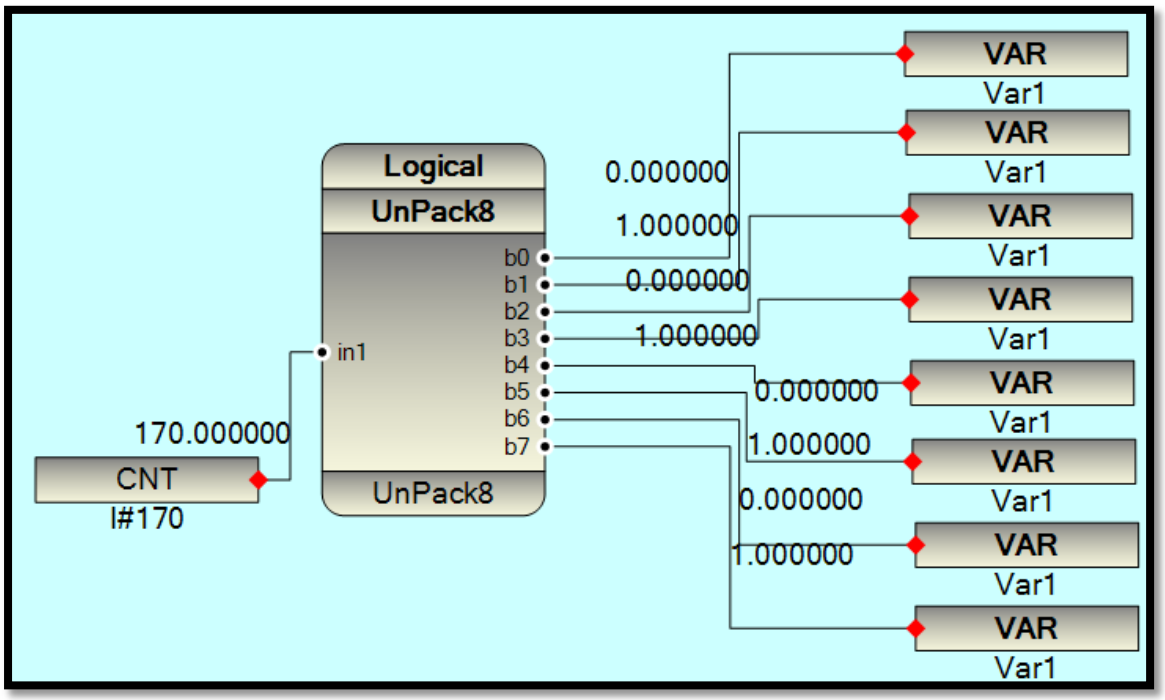


21.4.8 UnPack8 Function Block

UnPack8 will Convert one 8-bit integer to 8 digital signal (0 or 1)

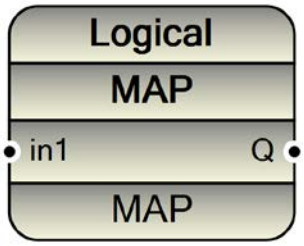
UnPack8			
	**I/O**	**Data Type**	**Description**
	In1	**Double**	*Input(Integer value)*
	b0,…, b7	**Double**	*Digital Output*

For Example: $\mathbf{170} = (\mathbf{10101010})_{\mathbf{2}}$

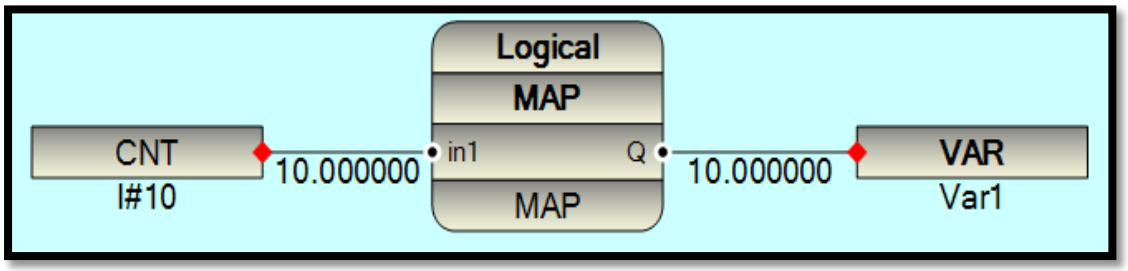


21.4.9 Map Function Block

Map Input signal to Output Signal

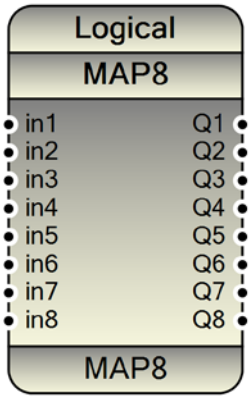
Map			
	**I/O**	**Data Type**	**Description**
	In1	**Double**	*Input signal*
	Q	**Double**	*Output signal*

For Example:

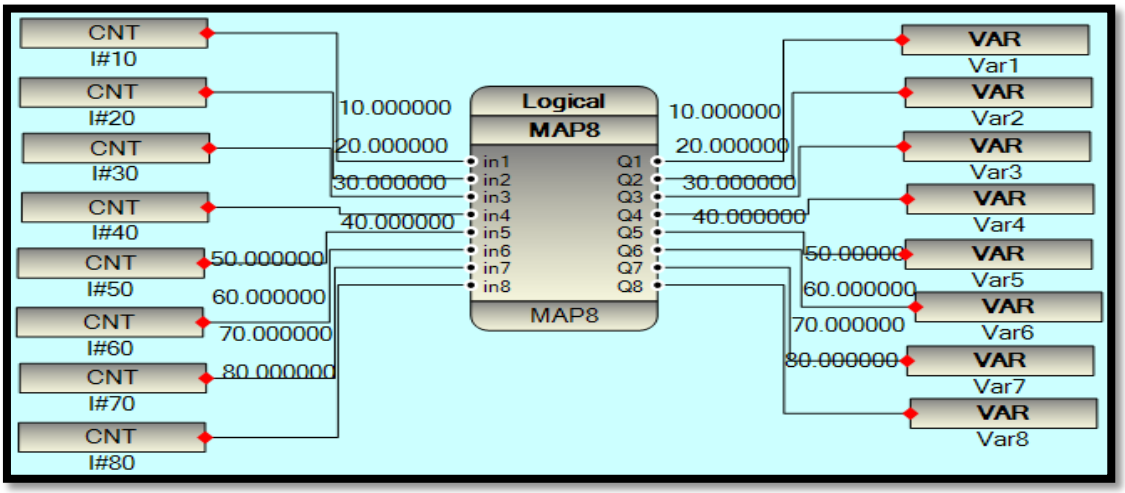


21.4.10 Map8 Function Block

Map 8 Input signal to 8 Output Signal

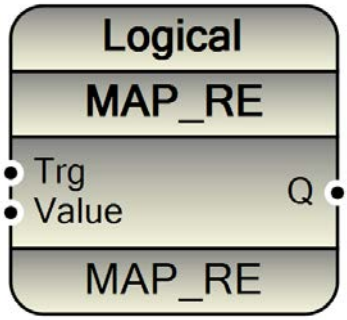
Map8			
	**I/O**	**Data Type**	**Description**
	In1	**Double**	*Input signal*
	Q	**Double**	*Output signal*

For Example:

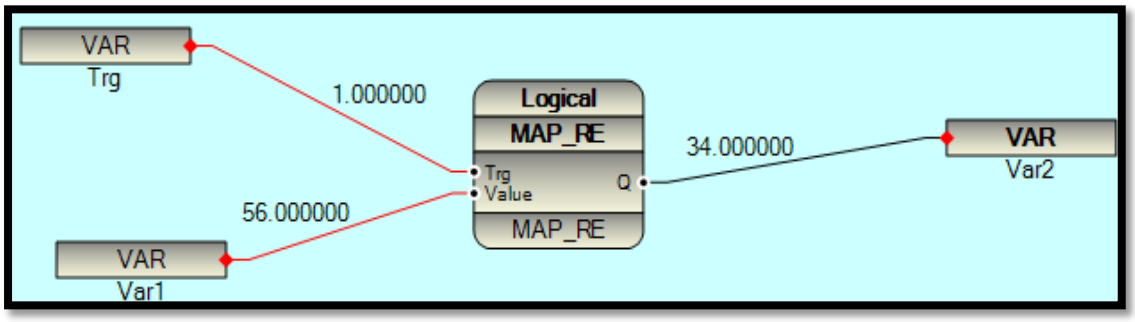


21.4.11 Map RE Function Block

Map value to Q when Trg changed from 0 to 1. When Input Value is changed, it will not map to Q until Trg is changed from 0 to 1.

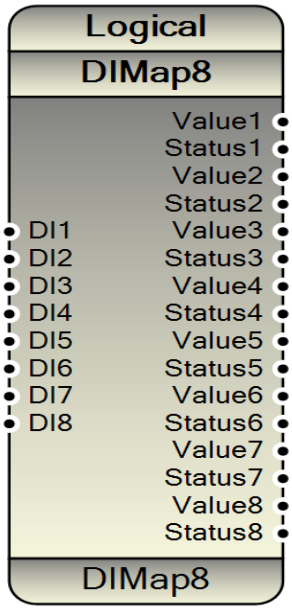
Map			
	**I/O**	**Data Type**	**Description**
	Trg	**Double**	*Trigger*
	Value	**Double**	*Input signal*
	Q	**Double**	*Output signal*

For Example:

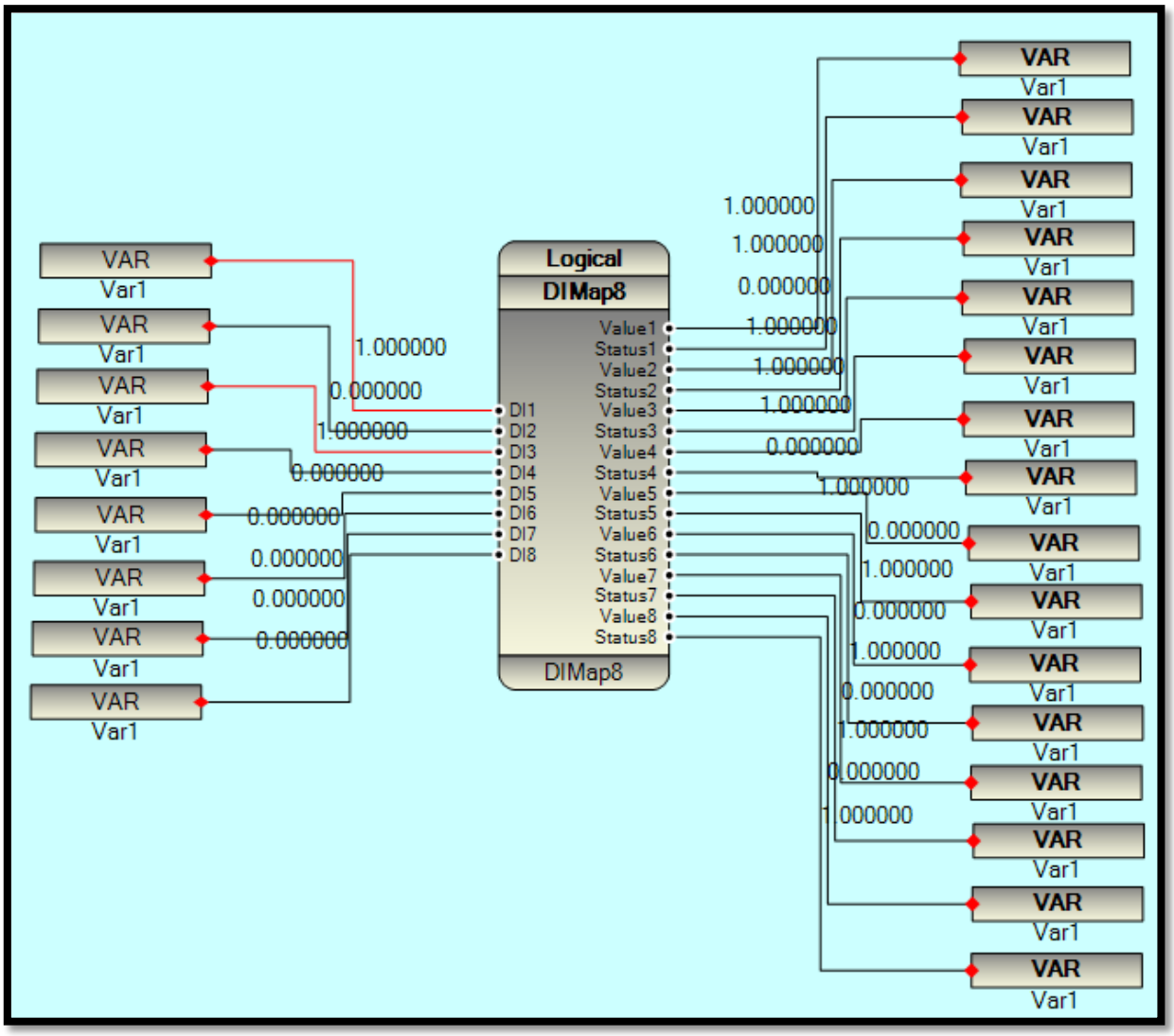


21.4.12 DIMap8 Function Block

when DI is stable, Status is 1 but when DI is unstable, get in chattering filter and Status is 32.
-Value is equal of DI.

DIMap8			
	**I/O**	**Data Type**	**Description**
	DI1,…., DI8	**Double**	*Digital Input*
	Value1, …, Value8	**Double**	*Digital Output*
	Status1, …, Status8	**Double**	*Status of input*

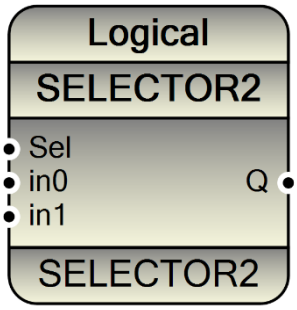
For Example:



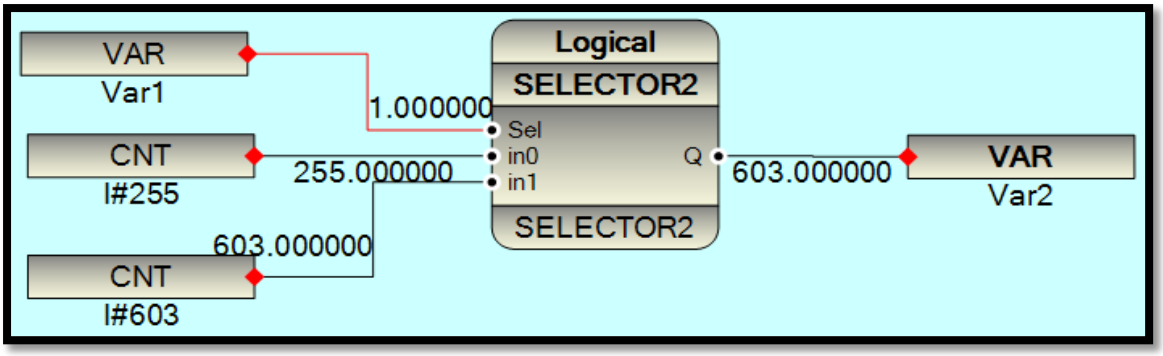
21.4.13 RIODIMap Function Block

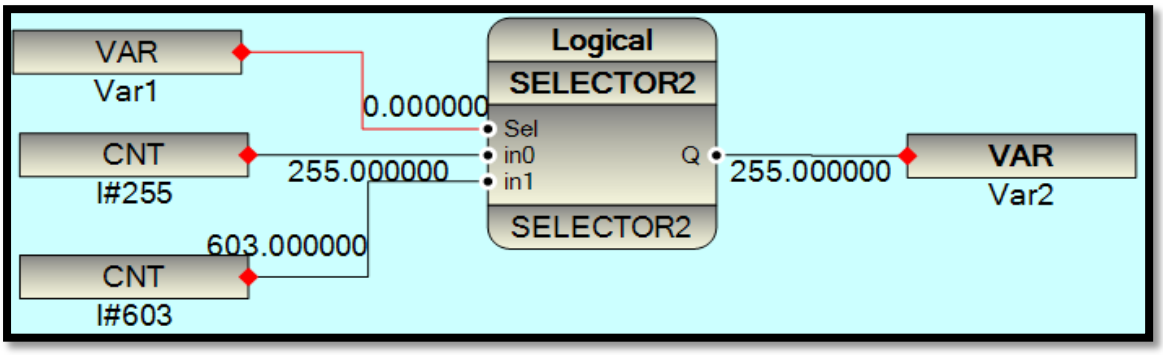
21.4.14 SEECTOR2 Function Block

When Sel Signal is 0, in0 is mapped to Q, When Sel is 1, in1 is mapped to Q.

SELECTOR2			
	**I/O**	**Data Type**	**Description**
	Sel	**Double**	*Selector*
	in0	**Double**	*Input 0*
	in1	**Double**	*Input 1*

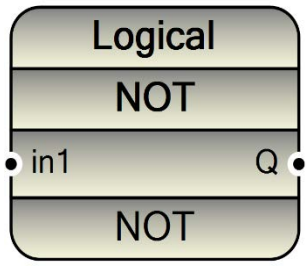
For Example:



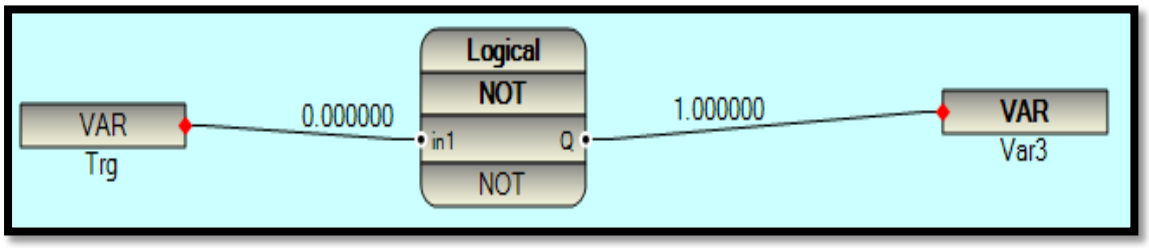


21.4.15 NOT Function Block

Reverse Function
When In1 is 0, Q is 1
When In1 is not 0, Q is 0

NOT			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*A*
	Q	**Double**	*A'*

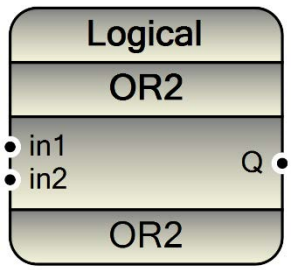
For Example:



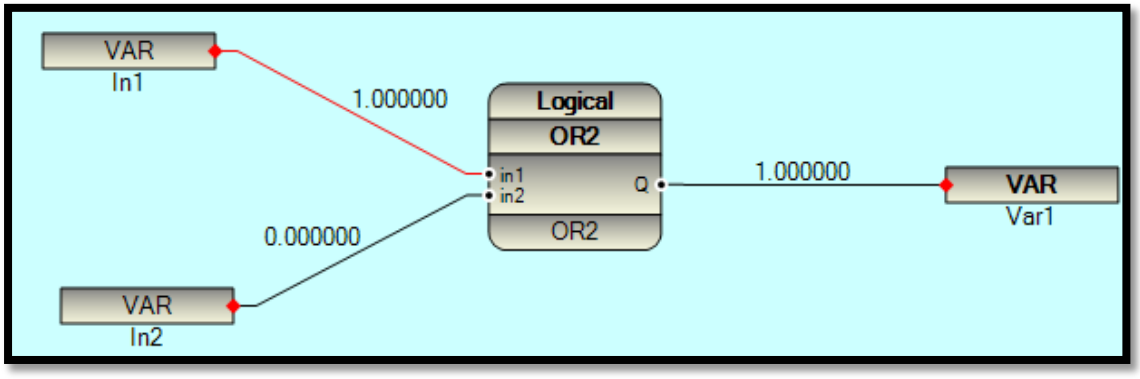
21.4.16 OR Function Block

OR2/OR3/OR4/OR5/OR6/OR7/OR8

Make OR all input signals and map to Output.

OR			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*A*
	in1	**Double**	*B*
	Q	**Double**	*A//B*

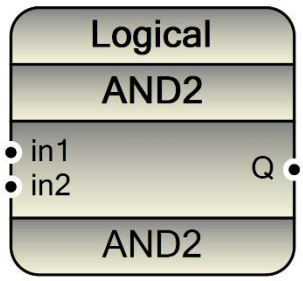
For Example:



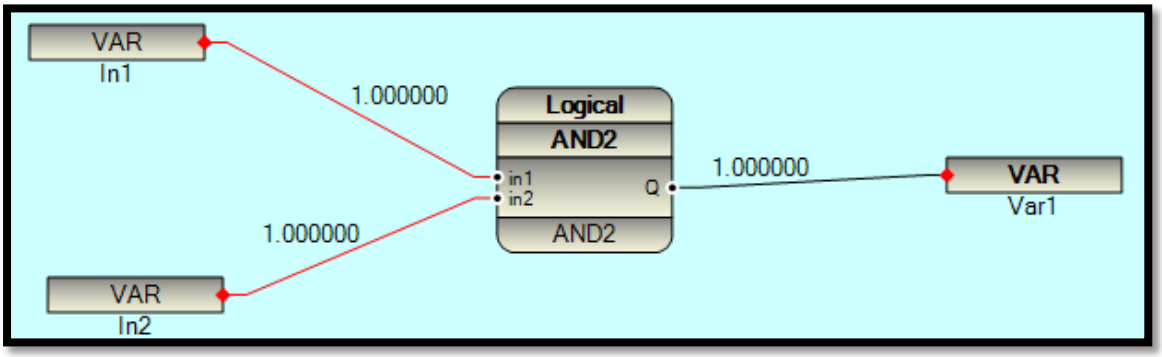
21.4.17 AND Function Block

AND2/AND3/AND4/AND5/AND6/AND7/AND8

Make AND all input signals and map to Output.

AND			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*A*
	in1	**Double**	*B*
	Q	**Double**	*A&B*

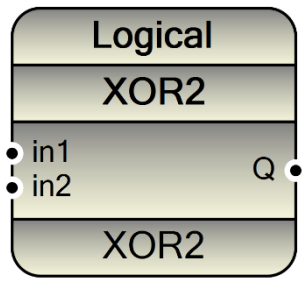
For Example:

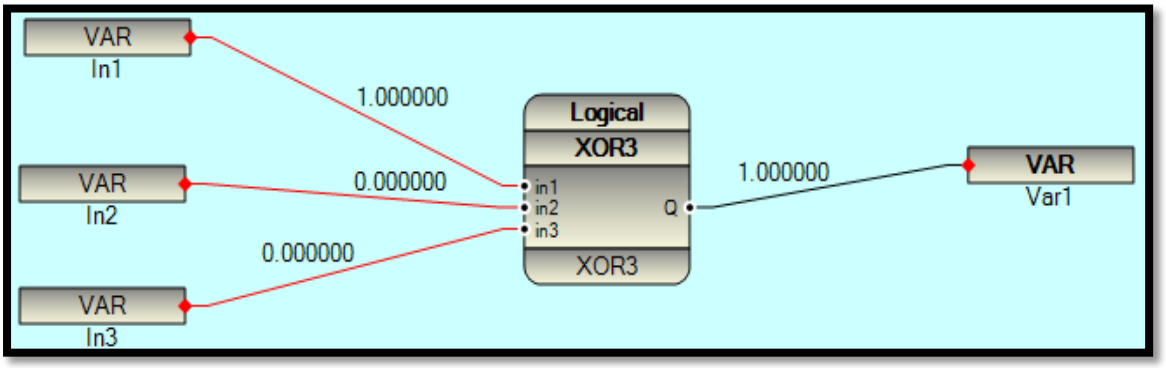


21.4.18 XOR Function Block

XOR2/XOR3/XOR4/XOR5/XOR6/XOR7/XOR8

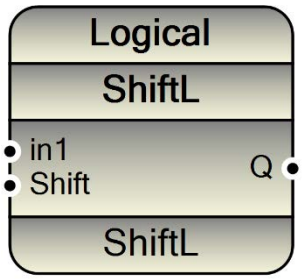
Make XOR all input signals and map to Output. When all Input Signals or just one input is 1, Q will set to 1 Otherwise Q is set to 0

XOR			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*A*
	in1	**Double**	*B*
	Q	**Double**	*A⊕B*

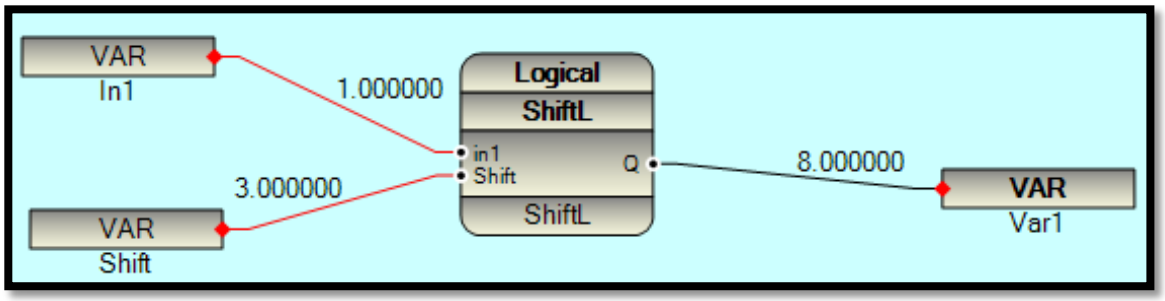


21.4.19 ShiftL Function Block

Shift Left. In1 Signals is shift to left side by Shift Number.

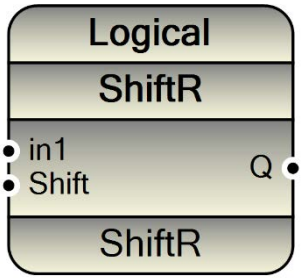
ShiftL			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*Initial Value*
	Shift	**Double**	*Shift*
	Q	**Double**	*Output*

For Example: $\mathbf{1 = (1)_2 \rightarrow^{ShiftL=3} \rightarrow (100)_2 = 8}$

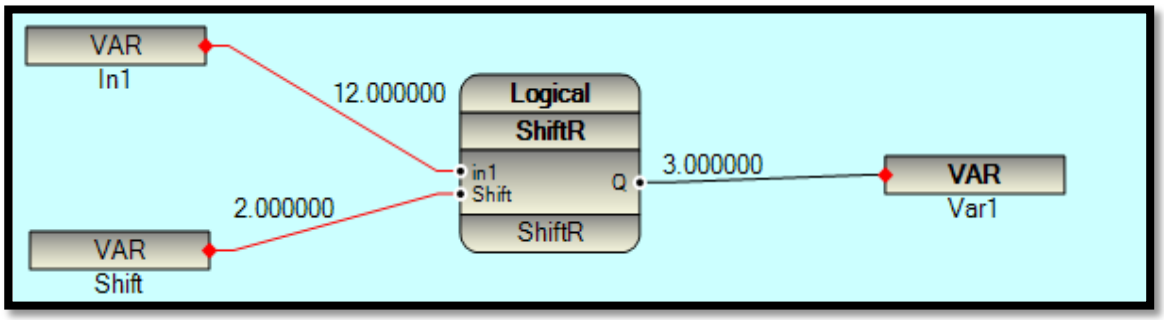


21.4.20 ShiftR Function Block

Shift Right. In1 Signals is shift to right side by Shift Number.

ShiftR			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*Initial Value*
	Shift	**Double**	*Shift*
	Q	**Double**	*Output*

For Example: $\mathbf{12 = (1100)_2 \rightarrow^{ShiftR=2} \rightarrow (11)_2 = 3}$

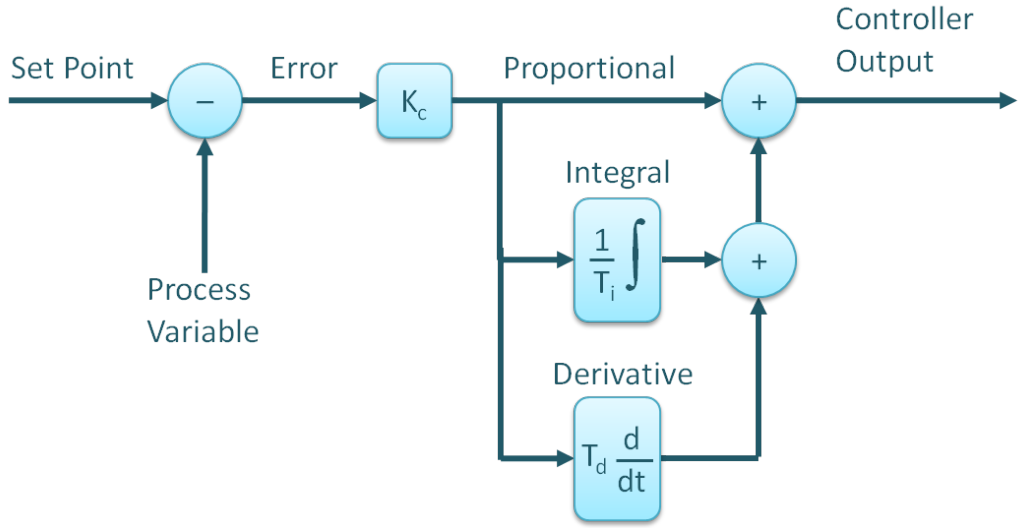


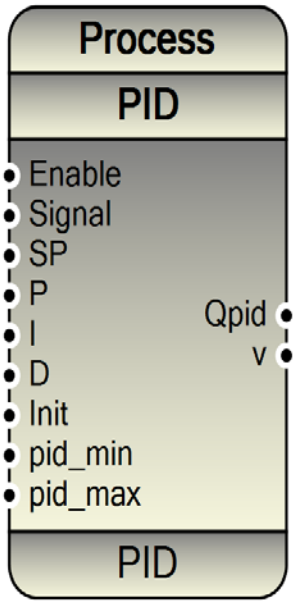
21.5 Process Group:

Process Function Groups

21.5.1 PID Function Block

This Function Block is Standard PID Function.



Integral			
	**I/O**	**Data Type**	**Description**
	Enabel	**Double**	*When Enable is 1, Function Block is Enable, otherwise Outputs are 0.*
	Signal	**Double**	*Input signal which should be controlled by FB.*
	SP	**Double**	*Set Point*
	P	**Double**	*Proportional parameter*
	I	**Double**	*Integral Parameter*
	D	**Double**	*Derivative Parameter*
	Init	**Double**	*Init (Bias) Value of Output*
	pid-min	**Double**	*Minimum Value of Output signal*
	pid-max	**Double**	*Maximum Value of Output Signal*
	Qpid	**Double**	*PID Output signal*
	V	**Double**	*Validity Signal. When Qpid Signal is between Pid_Max and Pid_Max, V is set to 1 (Output is Valid) otherwise it is set to 0*

Source Code

```
if(Enable==1)
{
     Error = SP - Signal;
     if (fabs(Error)<0.01)
     {
         Error = 0;
     }
    double delta = ErrorOld - Error;
    double DelaTime = (pbsgetTime() - TmpTimeOld)/1000; // sec
    double derivative = delta / DelaTime;
    double Integral = ((ErrorOld + Error)/2) * DelaTime +IntegralOld;
    IntegralOld = Integral;

    pidQ = (Error * P) + (Integral  * I) + (derivative * D) +Init;
    if((pidQ>PidMin) &&(pidQ<PidMax))
    {
        pidV = 1;
    }
    else
    {
        pidV = 0;
        if((pidQ<PidMin))
        {
            pidQ = PidMin;
        }
        else
        {
            if((pidQ>PidMax))
            {
                pidQ = PidMax;
             }
         }
     }

    ErrorOld = Error;
}// Enable
else
{
    pidQ = 0;
    pidV = 0;
}
```

Standard PID Tuning Methods

21.5.1.1 Cohen-Coon Method (Open-loop Test)

Step 1: Perform a step test to obtain the parameters of a FOPTD (first order plus time delay) model

i. *Make sure the process is at an initial steady state*
ii. *Introduce a step change in the manipulated variable*
iii. *Wait until the process settles at a new steady state*

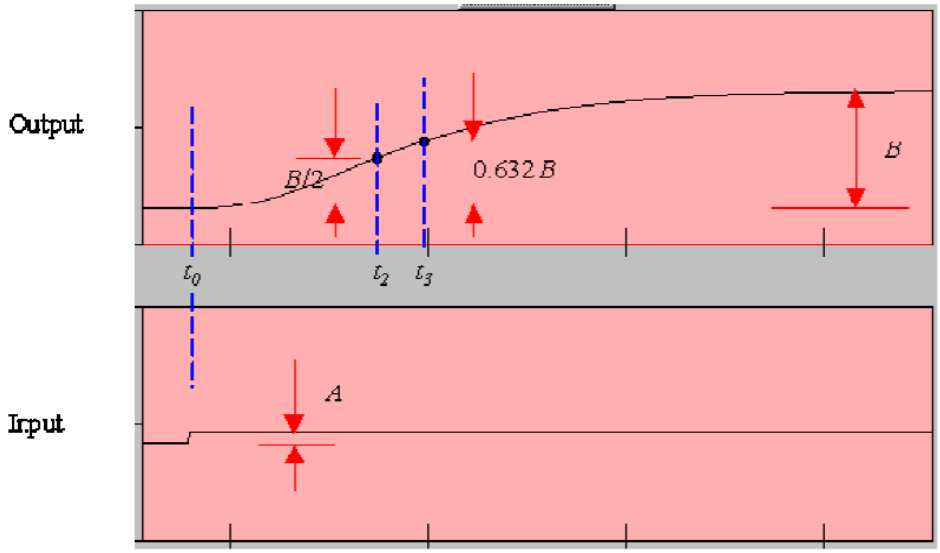


Figure 1. Step Test for Cohen-Coon Tuning.

Step 2: Calculate process parameters: $t_1, \tau, \tau_{del}, K, r$ *as follows*

$$t_1 = \frac{t_2 - (\ln(2))t_3}{1 - \ln(2)}$$
$$\tau = t_3 - t_1$$
$$\tau_{del} = t_1 - t_0$$
$$K = \frac{B}{A}$$
$$r = \frac{\tau_{del}}{\tau}$$

Step 3: Using the process parameters, use the prescribed values given by Cohen and Coon.

Table 1. Cohen-Coon Tuning Rules

	K_c	τ_{Int}	τ_{Der}
P	$\frac{1}{rK}\left(1+\frac{r}{3}\right)$		
PI	$\frac{1}{rK}\left(0.9+\frac{r}{12}\right)$	$\tau_{del}\frac{30+3r}{9+20r}$	
PID	$\frac{1}{rK}\left(\frac{4}{3}+\frac{r}{4}\right)$	$\tau_{del}\frac{32+6r}{13+8r}$	$\tau_{del}\frac{4}{11+2r}$

24.5.1.2 Ziegler-Nichols Method (Closed-loop P-Control Test)

Step 1: Determine the sign of process gain (e.g. open loop test as in Cohen-Coon).
Step 2: Implement a proportional control and introducing a new set-point.
Step 3: Increase proportional gain until sustained periodic oscillation.
Step 4: Record ultimate gain and ultimate period:K_u and P_u
Step 5: Evaluate control parameters as prescribed by Ziegler and Nichols

Table 2. Ziegler Nichols Tuning Rules

	K_c	τ_{Int}	τ_{Der}
P	$\frac{K_u}{2}$		
PI	$\frac{K_u}{2.2}$	$\frac{P_u}{1.2}$	
PID	$\frac{K_u}{1.7}$	$\frac{P_u}{2}$	$\frac{P_u}{8}$

24.5.1.3 Tyreus-Luyben Method (Closed-loop P-Control test)

Step 1-4: Same as steps 1 to 4 of Ziegler-Nichols method above
Step 5: Evaluate control parameters as prescribed by Tyreus and Luyben
Table 2. Tyreus-Luyben Tuning Rules for PI and PID

Table 2. Tyreus-Luyben Tuning Rules for PI and PID

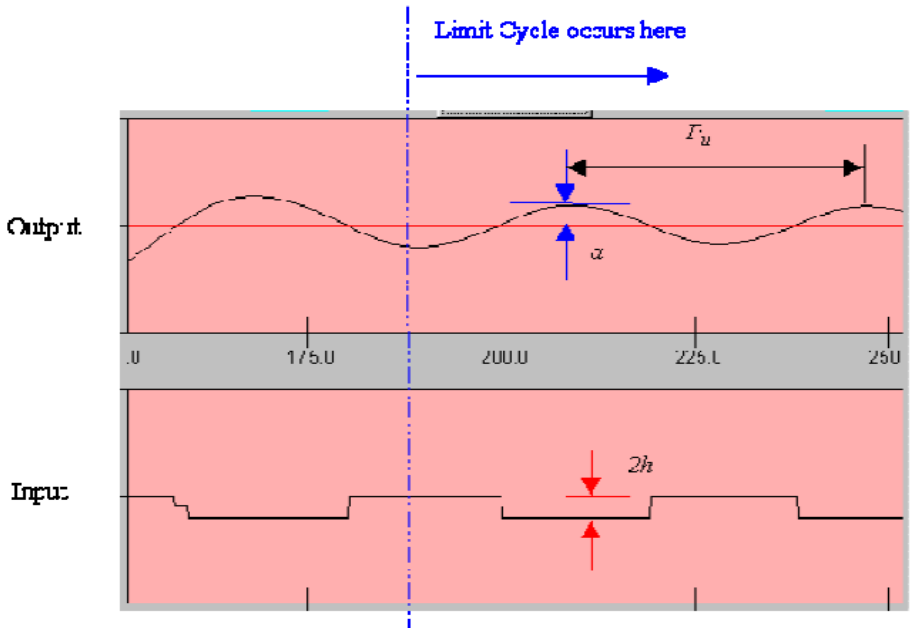
	K_c	τ_{Int}	τ_{Der}
PI	$\frac{K_u}{3.2}$	$2.2P_u$	
PID	$\frac{K_u}{2.2}$	$2.2P_u$	$\frac{P_u}{6.3}$

24.5.1.4 Auto tune Method (Closed-loop On-Off test)

Step 1: Let process settle to a steady state
Step 2: Move the SetPoint to the current steady state
Step 3: Implement an on-off (relay) controller

If process gain is positive, $u = \begin{cases} u_0 + h & if \ \ e \geq 0 \\ u_0 - h & if \ \ e < 0 \end{cases}$

Step 4: If process gain is negative, $u = \begin{cases} u_0 - h & if \ \ e \geq 0 \\ u_0 + h & if \ \ e < 0 \end{cases}$ *llation*

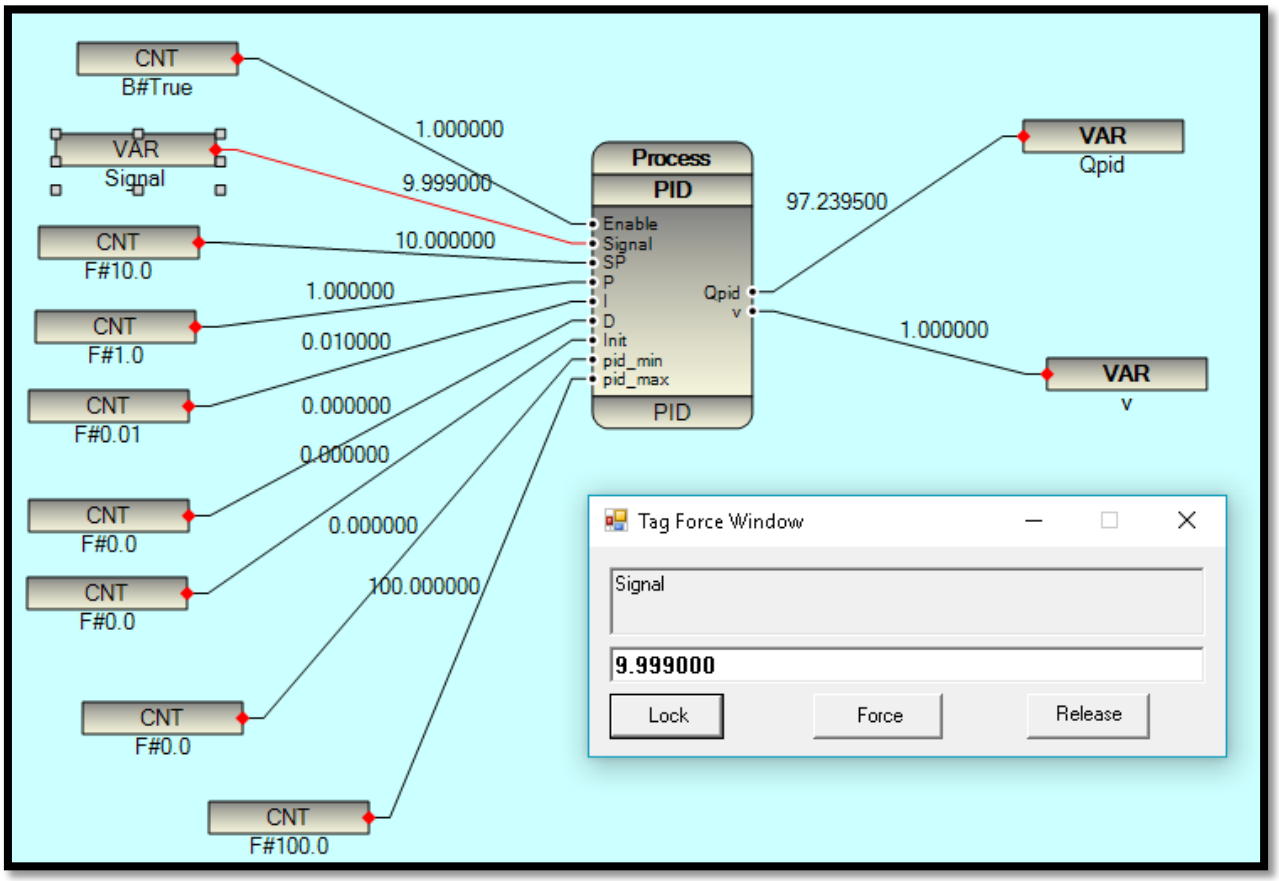


Step 5: Evaluate ultimate gain using auto tune formulas (-) can be obtain from the

$$K_u = \frac{4}{\pi}\frac{h}{a}$$

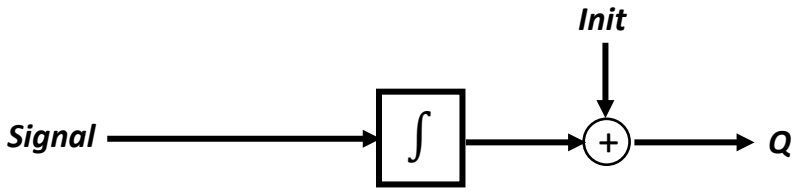
Step 6: Use either Ziegler-Nichols or Tyreus-Luyben prescribed tunings

For Example:

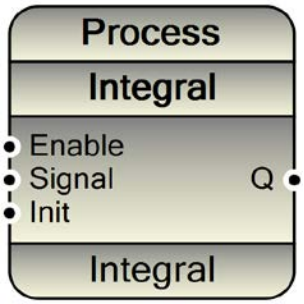


21.5.2 Integral Function Block

This Function Block is Standard Integral Function.

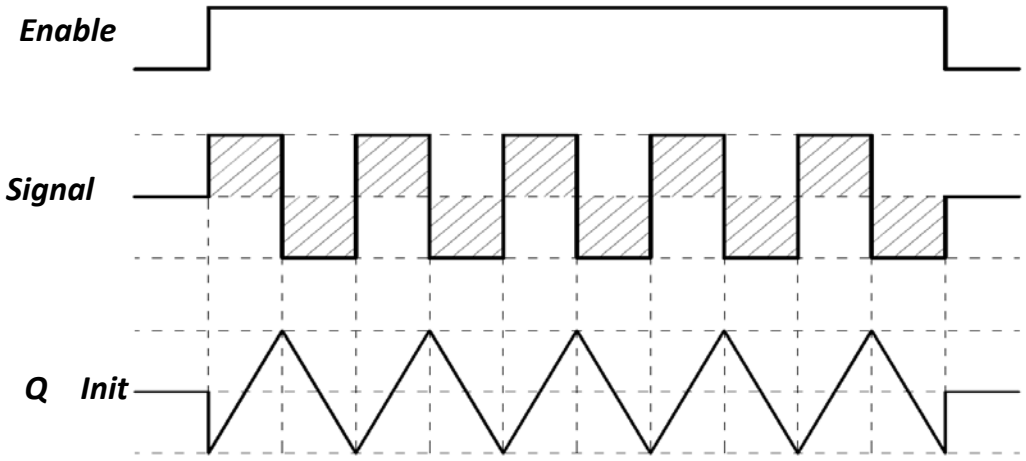


Integral



I/O	Data Type	Description
Enabel	Double	*When 1 , FB is enabled , Otherwise FB Output is 0*
Signal	Double	*Input Signal for getting integral*
Init	Double	*Init (Bias) Value for integration*
Q	Double	*Integrated Output signal*

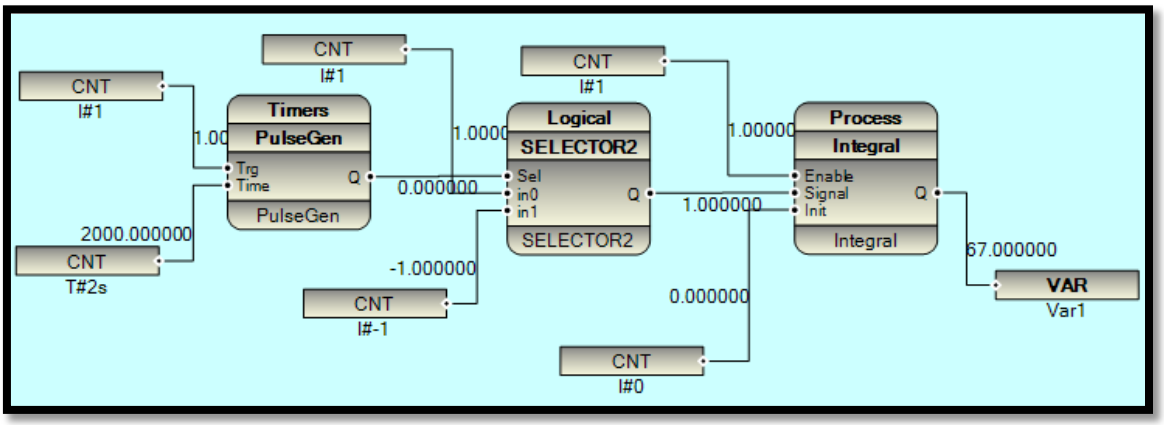
Example :Integral square pulse Diagram

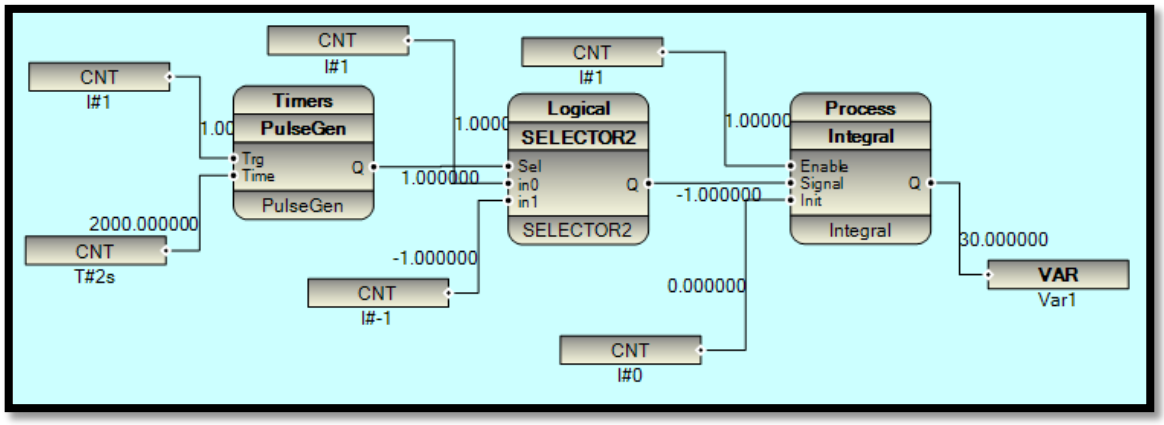


Source Code

```
if(Enable==1)
{

    double DelaTime = (pbsgetTime() - TmpTimeOld)/1000; // sec
    pidQ = ((SignalOld + Signal)/2) * DelaTime +IntegralOld;
    IntegralOld = pidQ;
    pidQ = pidQ+Init;
}// Enable
else
{
    pidQ = 0;
}
```

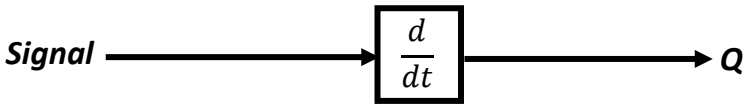
For Example:


CNT
I#1
CNT
I#1
CNT
I#1
Timers
PulseGen
Trg
Time
Q
PulseGen
1.00
2000.000000
CNT
T#2s
1.0000
0.000000
Logical
SELECTOR2
Sel
in0
in1
Q
SELECTOR2
-1.000000
CNT
I#-1
1.00000
Process
Integral
Enable
Signal
Init
Q
Integral
1.000000
0.000000
CNT
I#0
67.000000
VAR
Var1

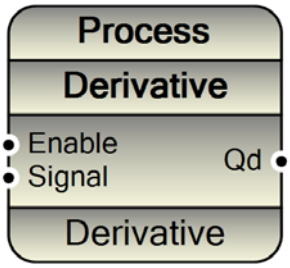

CNT
I#1
CNT
I#1
CNT
I#1
Timers
PulseGen
Trg
Time
Q
PulseGen
1.00
2000.000000
CNT
T#2s
1.0000
1.000000
Logical
SELECTOR2
Sel
in0
in1
Q
SELECTOR2
-1.000000
CNT
I#-1
1.00000
Process
Integral
Enable
Signal
Init
Q
Integral
-1.000000
0.000000
CNT
I#0
30.000000
VAR
Var1

21.5.3 Derivative Function Block

This Function Block is Standard Derivative Function.

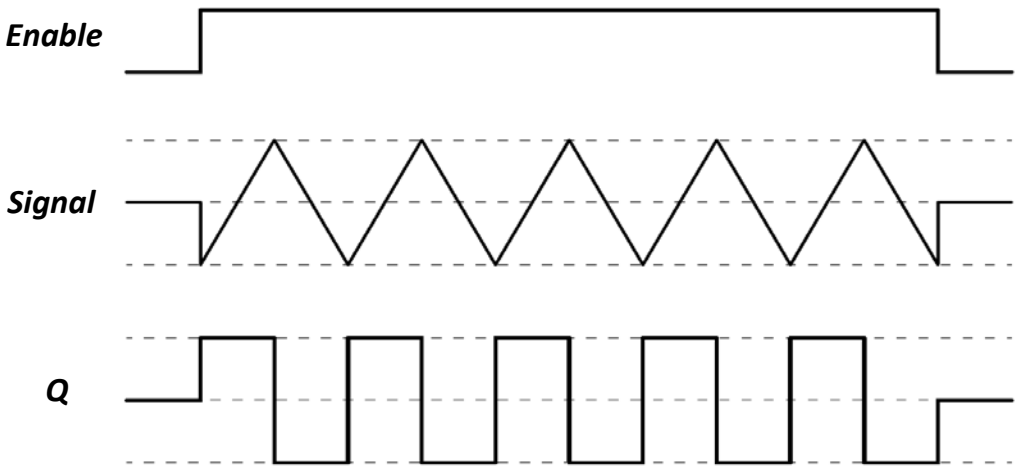


Integral



I/O	Data Type	Description
Enabel	Double	*When 1 , FB is enabled , Otherwise FB Output is 0*
Signal	Double	*Input Signal for getting Derivative*
Q	Double	*Derivative Output signal*

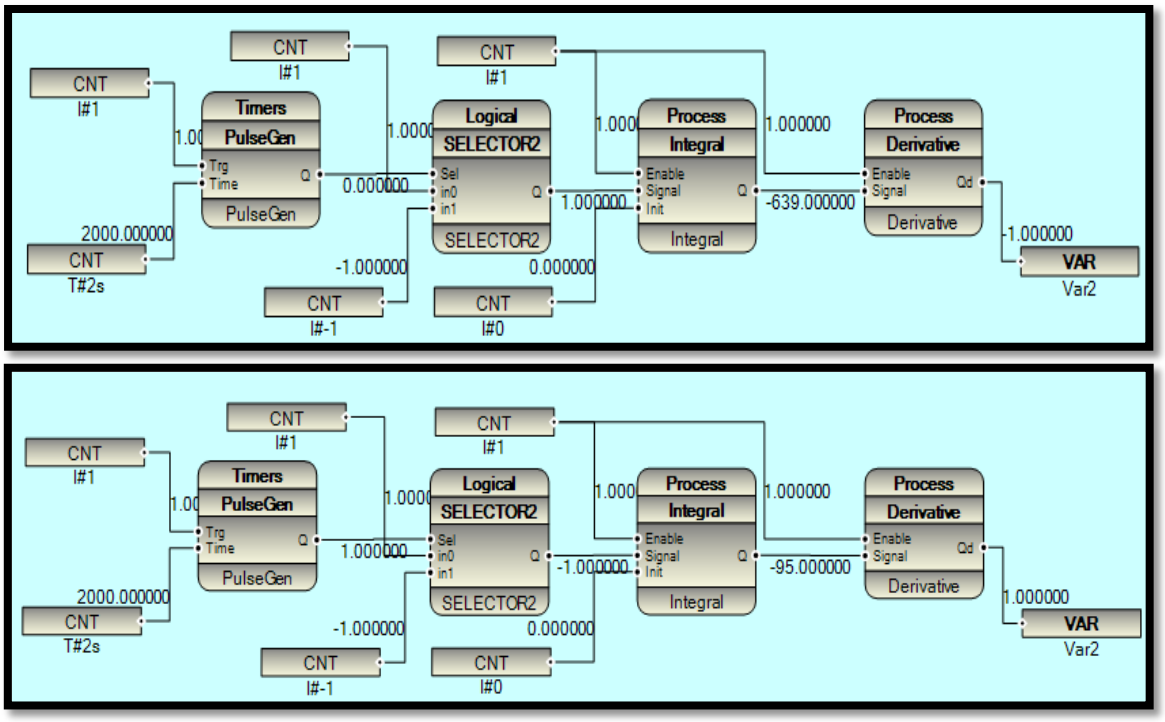
Example :Integral square pulse Diagram



Source Code

```
if(Enable==1)
{


    double DelaTime = (pbsgetTime() - TmpTimeOld)/1000; // sec
    double Delta = SignalOld - Signal ;
    pidQ = Delta / DelaTime;
}// Enable
else
{
    pidQ = 0;
}
```

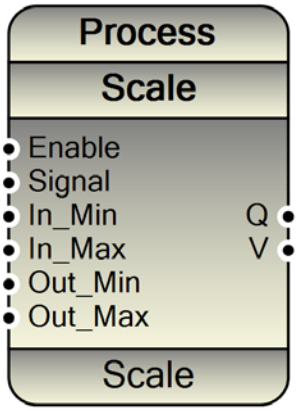
For Example:


CNT
I#1
CNT
I#1
CNT
I#1
Timers
PulseGen
Trg
Time
Q
PulseGen
Logical
SELECTOR2
Sel
in0
in1
Q
SELECTOR2
Process
Integral
Enable
Signal
Init
Q
Integral
Process
Derivative
Enable
Signal
Qd
Derivative
1.000000
VAR
Var2
2000.000000
CNT
T#2s
-1.000000
CNT
I#-1
0.000000
CNT
I#0
1.000000
-639.000000
1.000000
0.000000
1.000000
1.000000
CNT
I#1
CNT
I#1
CNT
I#1
Timers
PulseGen
Trg
Time
Q
PulseGen
Logical
SELECTOR2
Sel
in0
in1
Q
SELECTOR2
Process
Integral
Enable
Signal
Init
Q
Integral
Process
Derivative
Enable
Signal
Qd
Derivative
1.000000
VAR
Var2
2000.000000
CNT
T#2s
-1.000000
CNT
I#-1
0.000000
CNT
I#0
1.000000
-95.000000
1.000000
-1.000000
1.000000

21.5.4 Scale Function Block

This Function Block is Scaling Input Signal.

Scale

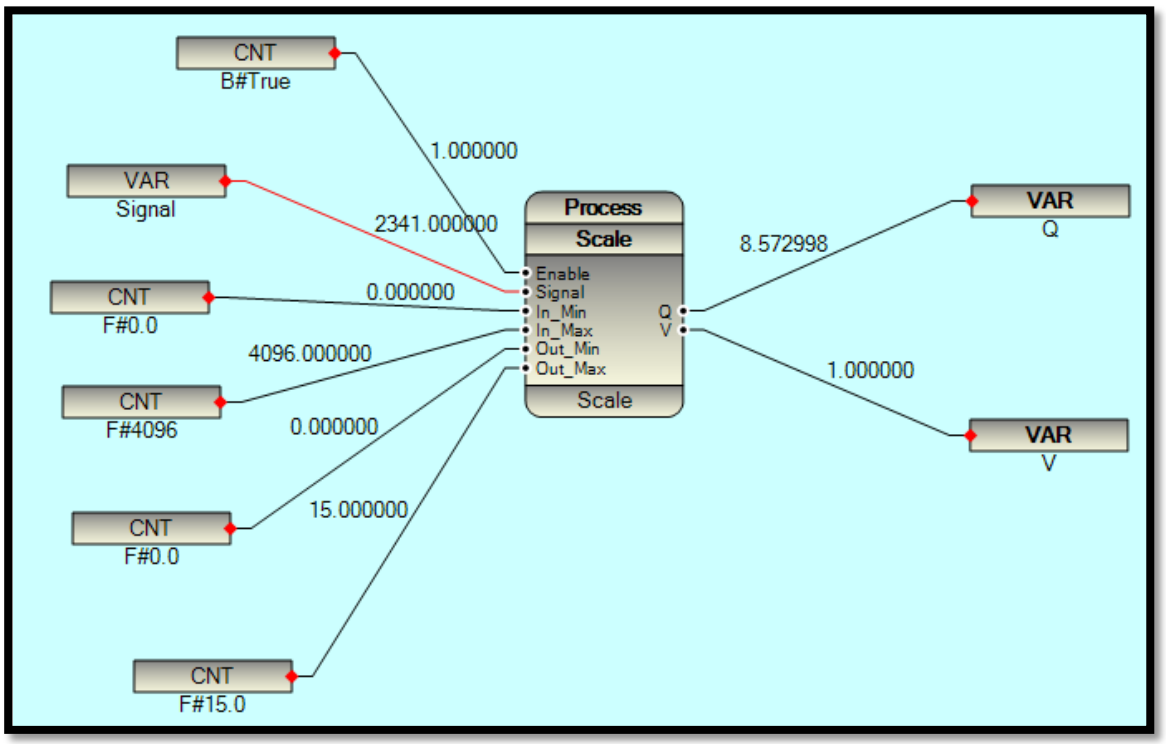


I/O	Data Type	Description
Enabel	Double	*When 1, FB is enabled, Otherwise FB Output is 0*
Signal	Double	*Raw Input Signal*
In_Min	Double	*Minimum Raw Value*
In_Min	Double	*Maximum Raw Value*
Out_Min	Double	*Minimum Value of scaled value*
Out_Min	Double	*Maximum Value of Scaled Value*
Q	Double	*Scaled Output signal*
V	Double	*Validity. When Output Signal is between Out_Min and Out_Max, V is set to 1, otherwise V is Set to 0*

Source Code

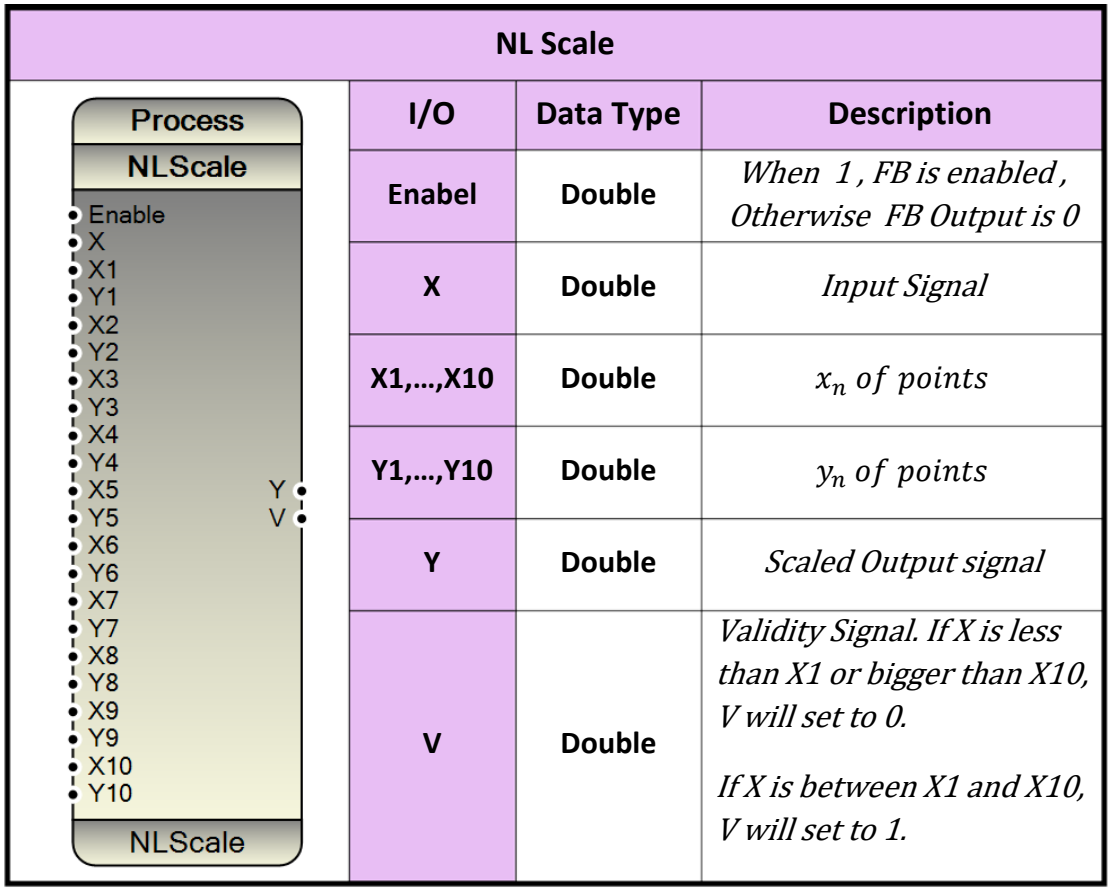
```
    TmpOut =Out_Min+((Out_Max -Out_Min )/(In_Max -In_Min ))*(TmpSignal-In_Min);

    if((TmpOut>=Out_Min) &&(TmpOut<=Out_Max))
    {
        TmpValid = 1;
    }
    else
    {
        TmpValid = 0;
        if(TmpOut>Out_Max)
        {
            TmpOut = Out_Max;

        }
        if(TmpOut<Out_Min)
        {
            TmpOut = Out_Min;
        }

    }
}
else
{
    TmpOut = 0;
    TmpValid = 0;

}
```

For Example:


CNT
B#True
1.000000
VAR
Signal
2341.000000
Process
Scale
Enable
Signal
In_Min
In_Max
Out_Min
Out_Max
Q
V
Scale
8.572998
VAR
Q
CNT
F#0.0
0.000000
4096.000000
CNT
F#4096
0.000000
1.000000
VAR
V
CNT
F#0.0
15.000000
CNT
F#15.0

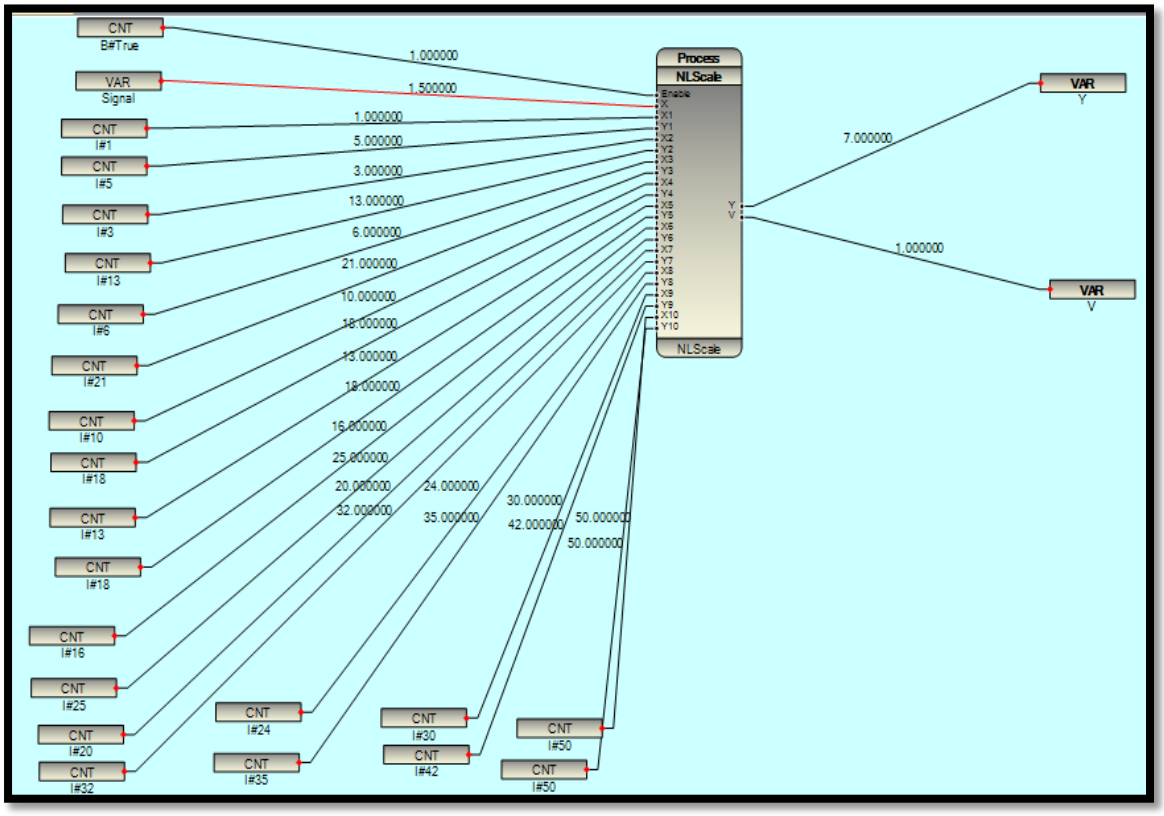
21.5.5 NL Scale Function Block

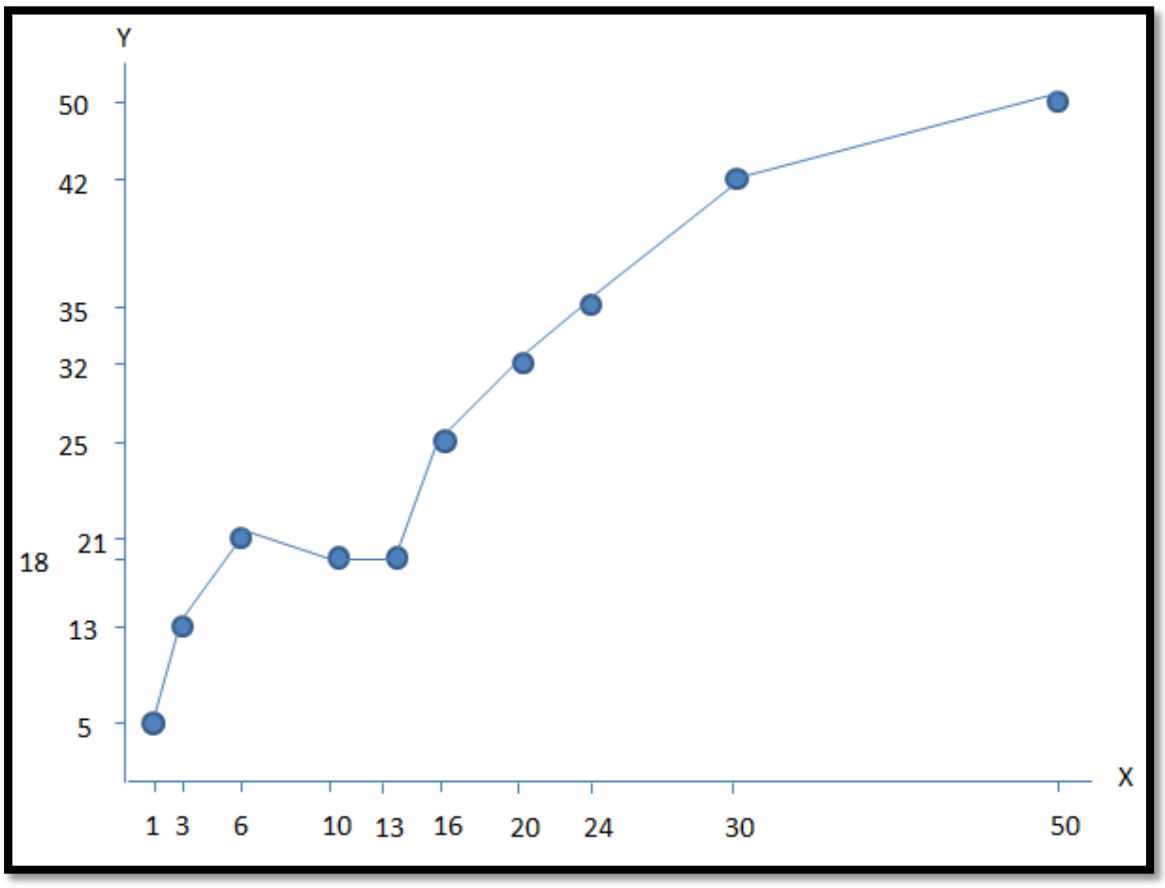
This Function Block is Nonlinear Scaling FB. you can model your scaling graph by 10 points as an example suppose we want to scale a signal based on following diagram:



NL Scale		
I/O	**Data Type**	**Description**
Enabel	**Double**	*When 1 , FB is enabled , Otherwise FB Output is 0*
X	**Double**	*Input Signal*
X1,…,X10	**Double**	x_n *of points*
Y1,…,Y10	**Double**	y_n *of points*
Y	**Double**	*Scaled Output signal*
V	**Double**	*Validity Signal. If X is less than X1 or bigger than X10, V will set to 0.* *If X is between X1 and X10, V will set to 1.*

For Example:

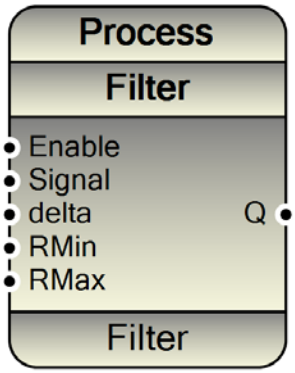



Y
50
42
35
32
25
21
18
13
5
X
1 3 6 10 13 16 20 24 30 50

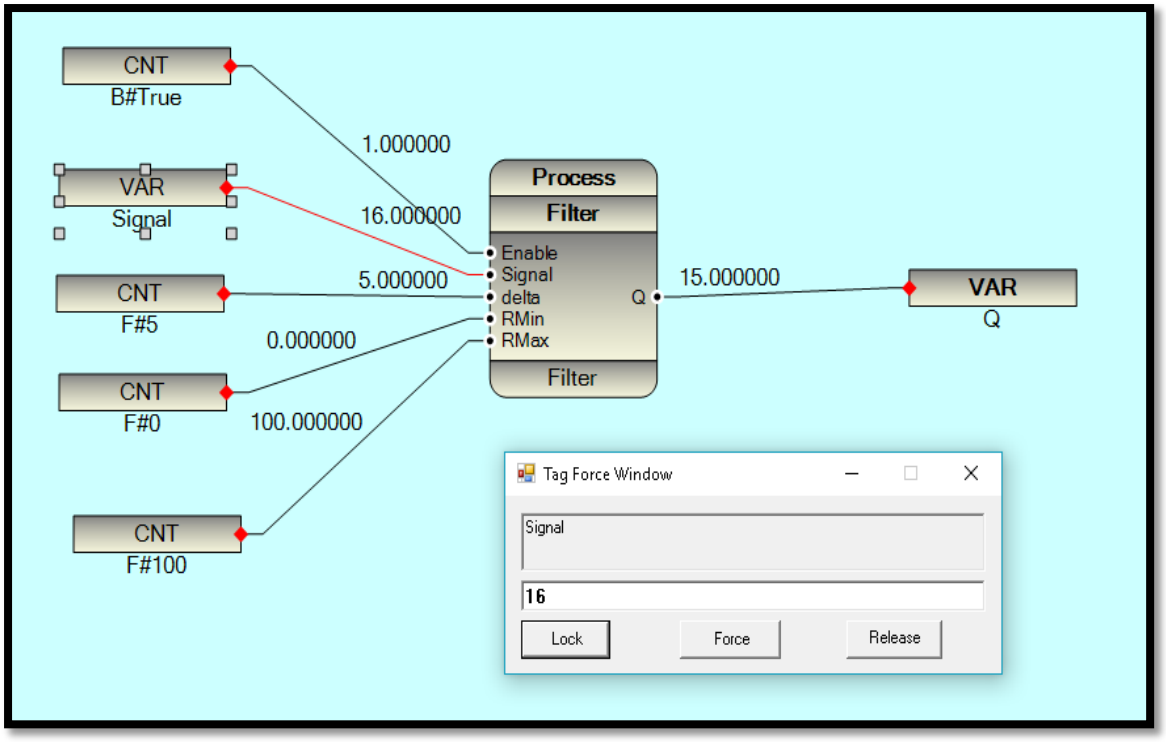
21.5.6 Filter Function Block

This Function Block is Standard Digital Filter.

In above example when Input signal changes is more than 5% then Input Signal will map to Output Signal, otherwise old value will pass to output.

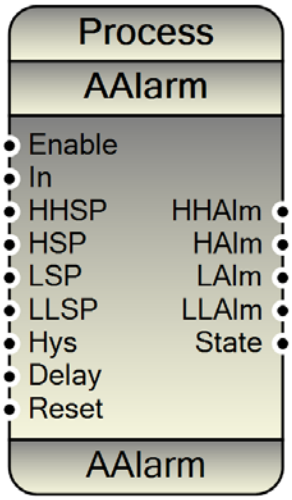
Filter			
	**I/O**	**Data Type**	**Description**
	Enabel	**Double**	*When Set to 1 , FB is operational*
	Signal	**Double**	*Input Signal*
	delta	**Double**	*When Input Signal Changes is more than Delta (in percentage) then Current value of Signal will pass to Q Output .*
	RMin	**Double**	*Minimum Range of Input Signal*
	RMax	**Double**	*Maximum Range of Input Signal*
	Q	**Double**	*Output*

For Example:

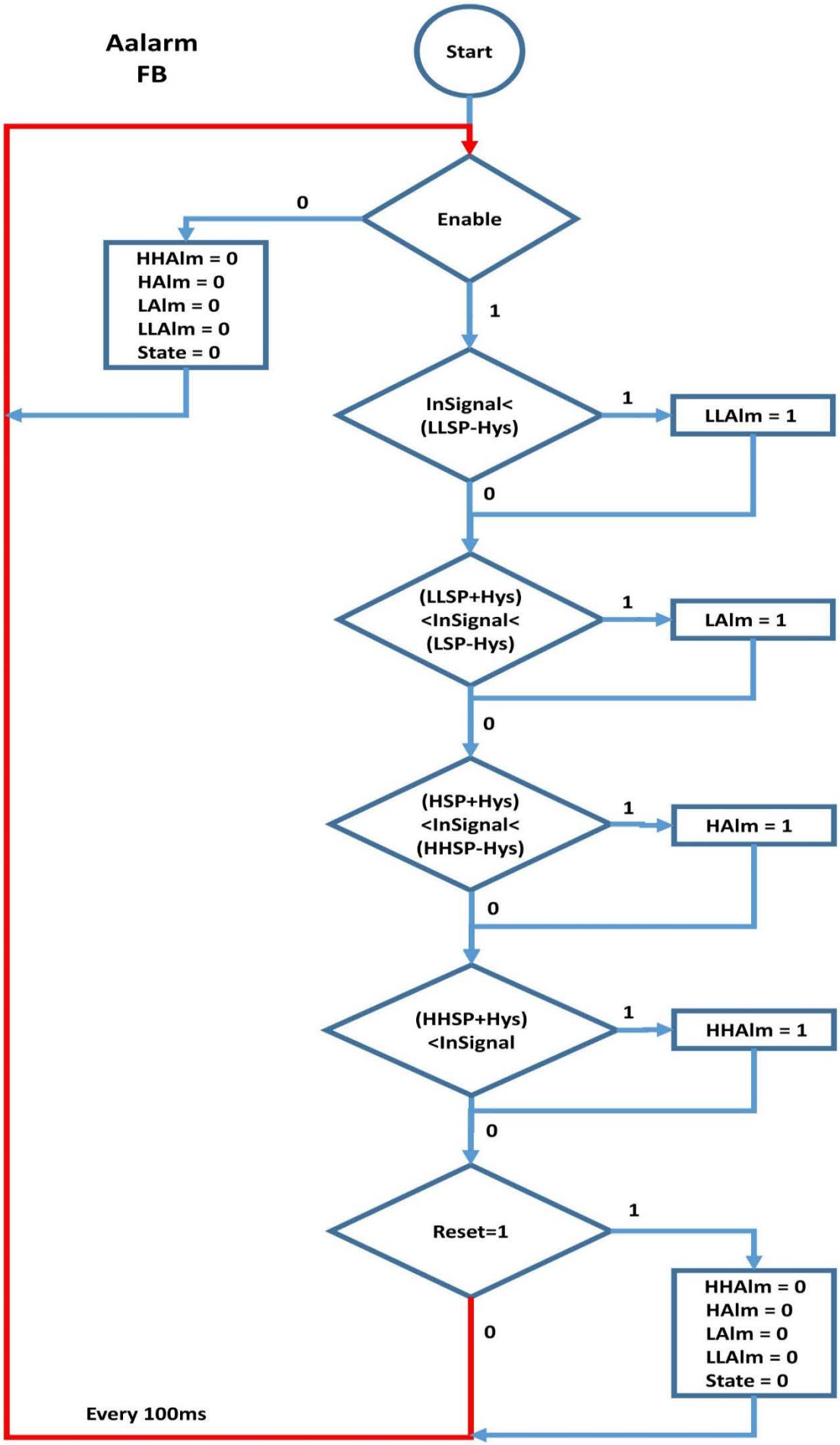


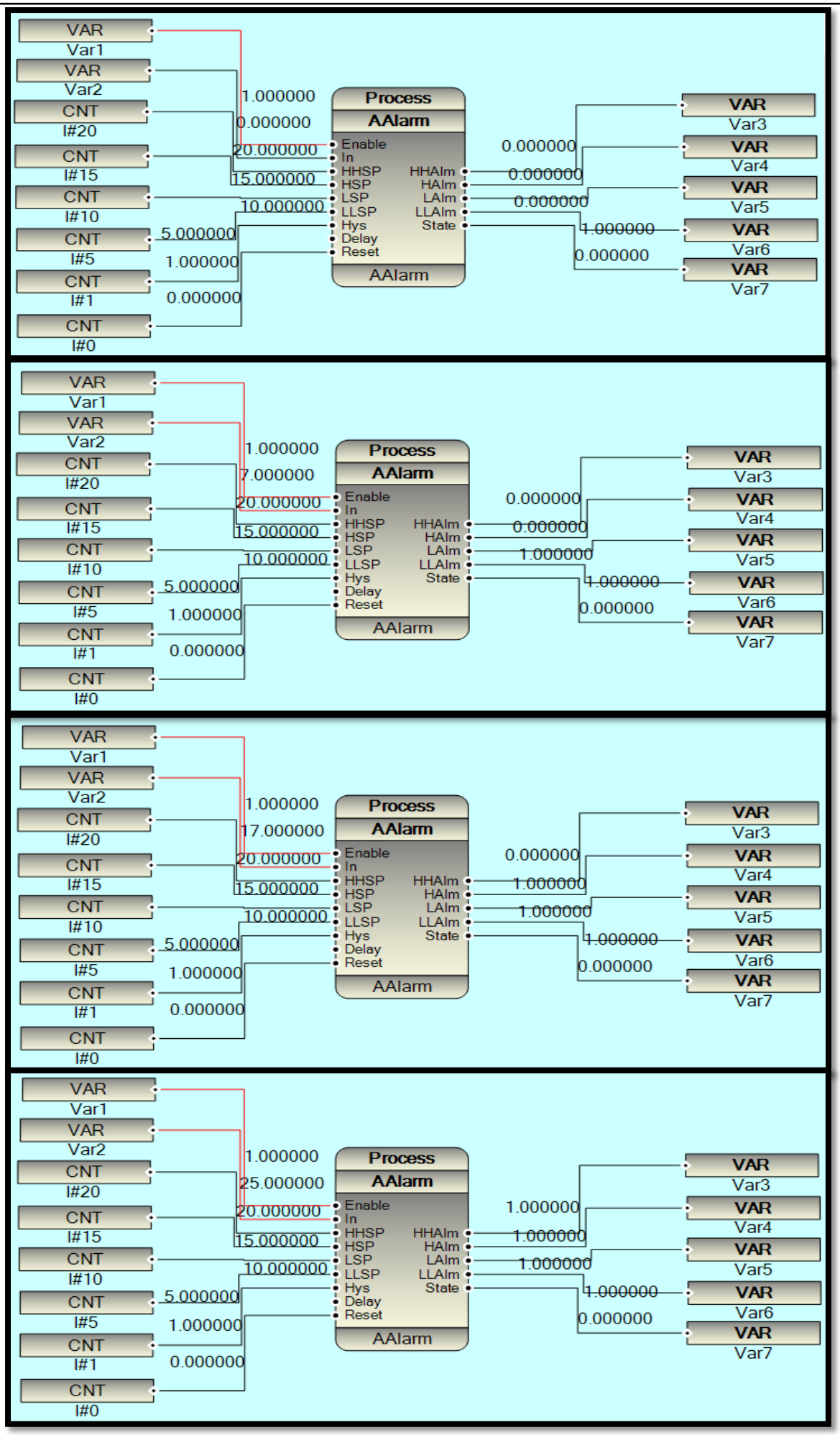
- **21.5.7 Driver1 Function Block**
- **21.5.8 Driver1V2 Function Block**
- **21.5.9 Driver2 Function Block**

21.5.10 AAlarm Function Block



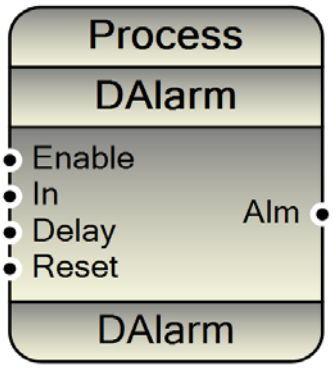
Filter		
I/O	**Data Type**	**Description**
Enable	**Double**	*Enable/Disable*
In	**Double**	*Input Signal*
HHSP	**Double**	*Maximum line*
HSP	**Double**	*Upper line*
LSP	**Double**	*Downer line*
LLSP	**Double**	*Minimum line*
Hys	**Double**	*Period*
Delay	**Double**	*NC*
Reset	**Double**	*Reset*
HHAlm	**Double**	*Maximum line Alarm*
HAlm	**Double**	*Upper line Alarm*
LAlm	**Double**	*Downer line Alarm*
LLAlm	**Double**	*Minimum line Alarm*
State	**Double**	*NC*


Aalarm
FB
Start
Enable
0
HHAlm = 0
HAlm = 0
LAlm = 0
LLAlm = 0
State = 0
1
InSignal<
(LLSP-Hys)
1
LLAlm = 1
0
(LLSP+Hys)
<InSignal<
(LSP-Hys)
1
LAlm = 1
0
(HSP+Hys)
<InSignal<
(HHSP-Hys)
1
HAlm = 1
0
(HHSP+Hys)
<InSignal
1
HHAlm = 1
0
Reset=1
1
HHAlm = 0
HAlm = 0
LAlm = 0
LLAlm = 0
State = 0
0
Every 100ms


VAR
Var1
VAR
Var2
CNT
I#20
CNT
I#15
CNT
I#10
CNT
I#5
CNT
I#1
CNT
I#0
1.000000
0.000000
20.000000
15.000000
10.000000
5.000000
1.000000
0.000000
Process
AAlarm
Enable
In
HHSP
HSP
LSP
LLSP
Hys
Delay
Reset
HHAlm
HAlm
LAlm
LLAlm
State
AAlarm
0.000000
0.000000
0.000000
1.000000
0.000000
VAR
Var3
VAR
Var4
VAR
Var5
VAR
Var6
VAR
Var7
VAR
Var1
VAR
Var2
CNT
I#20
CNT
I#15
CNT
I#10
CNT
I#5
CNT
I#1
CNT
I#0
1.000000
7.000000
20.000000
15.000000
10.000000
5.000000
1.000000
0.000000
Process
AAlarm
Enable
In
HHSP
HSP
LSP
LLSP
Hys
Delay
Reset
HHAlm
HAlm
LAlm
LLAlm
State
AAlarm
0.000000
0.000000
1.000000
1.000000
0.000000
VAR
Var3
VAR
Var4
VAR
Var5
VAR
Var6
VAR
Var7
VAR
Var1
VAR
Var2
CNT
I#20
CNT
I#15
CNT
I#10
CNT
I#5
CNT
I#1
CNT
I#0
1.000000
17.000000
20.000000
15.000000
10.000000
5.000000
1.000000
0.000000
Process
AAlarm
Enable
In
HHSP
HSP
LSP
LLSP
Hys
Delay
Reset
HHAlm
HAlm
LAlm
LLAlm
State
AAlarm
0.000000
1.000000
1.000000
1.000000
0.000000
VAR
Var3
VAR
Var4
VAR
Var5
VAR
Var6
VAR
Var7
VAR
Var1
VAR
Var2
CNT
I#20
CNT
I#15
CNT
I#10
CNT
I#5
CNT
I#1
CNT
I#0
1.000000
25.000000
20.000000
15.000000
10.000000
5.000000
1.000000
0.000000
Process
AAlarm
Enable
In
HHSP
HSP
LSP
LLSP
Hys
Delay
Reset
HHAlm
HAlm
LAlm
LLAlm
State
AAlarm
1.000000
1.000000
1.000000
1.000000
0.000000
VAR
Var3
VAR
Var4
VAR
Var5
VAR
Var6
VAR
Var7

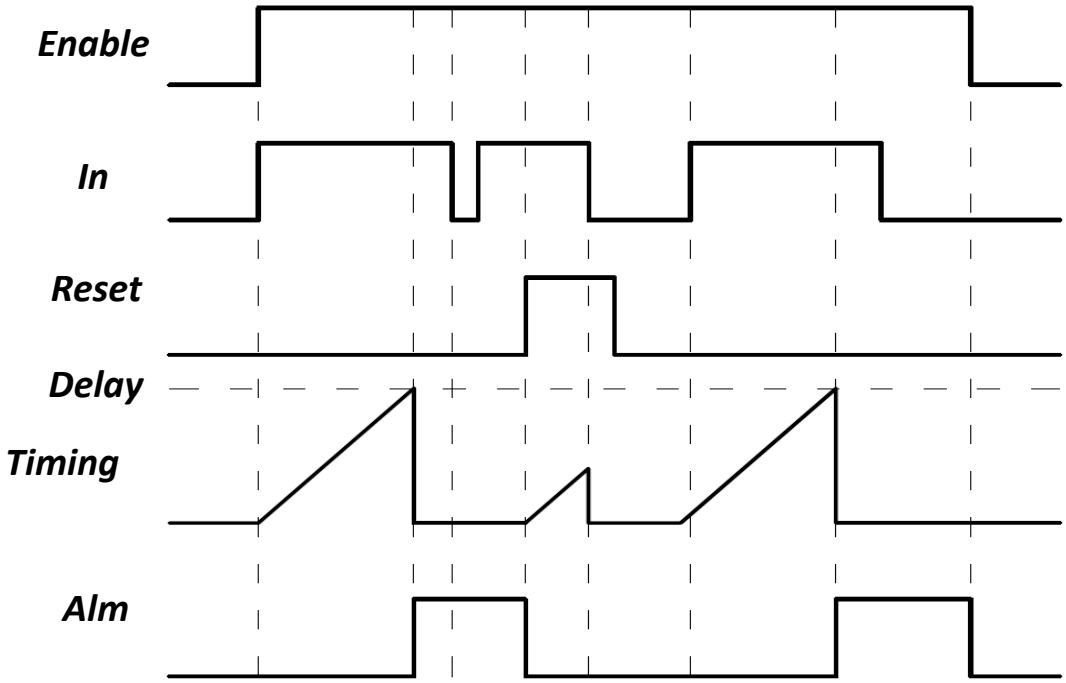
21.5.11 DAlarm Function Block

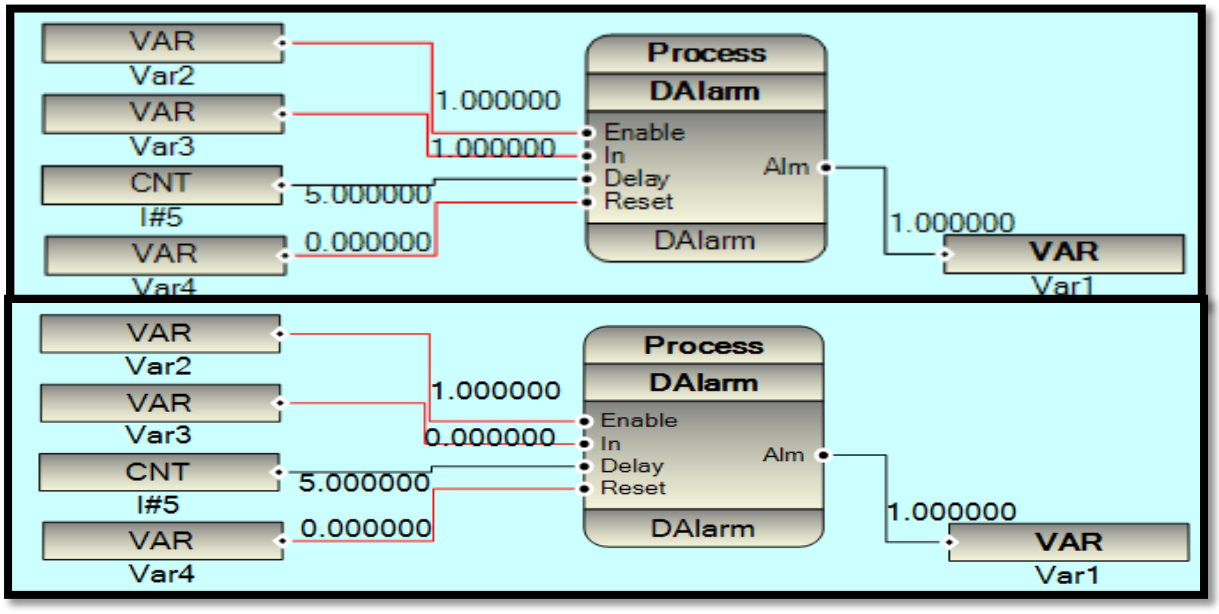
DAlarm



I/O	Data Type	Description
Enable	Double	*Enable/Disable*
In	Double	*Trigger*
Delay	Double	*Delay Time*
Reset	Double	*Reset*
Alm	Double	*Alarm*

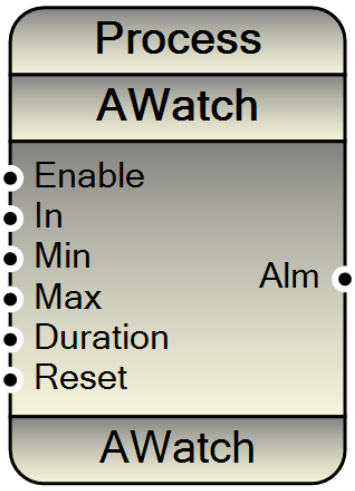
Timing Diagram





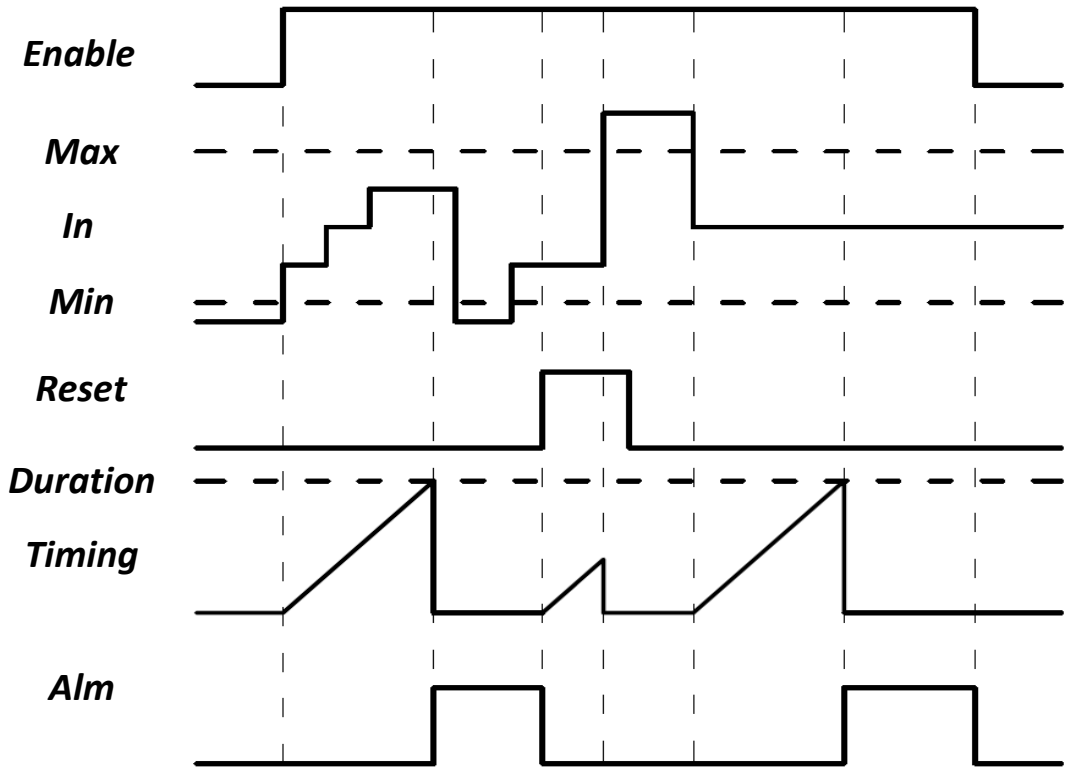
21.5.12 AWatch Function Block

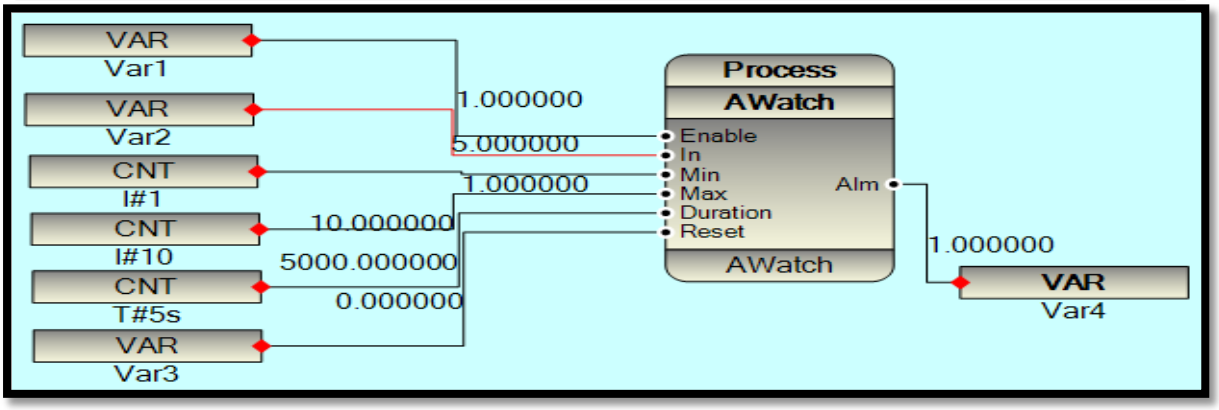
DAlarm



I/O	Data Type	Description
Enable	Double	*Enable/Disable*
In	Double	*Input value*
Min	Double	*Min value*
Max	Double	*Max Value*
Duration	Double	*Duration*
Reset	Double	*Reset*
Alm	Double	*Alarm*

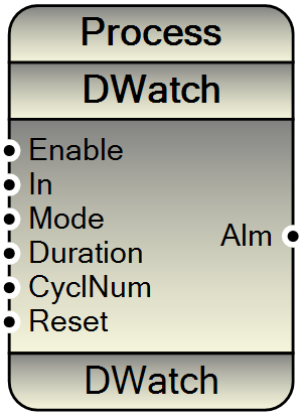
Timing Diagram





21.5.13 DWatch Function Block

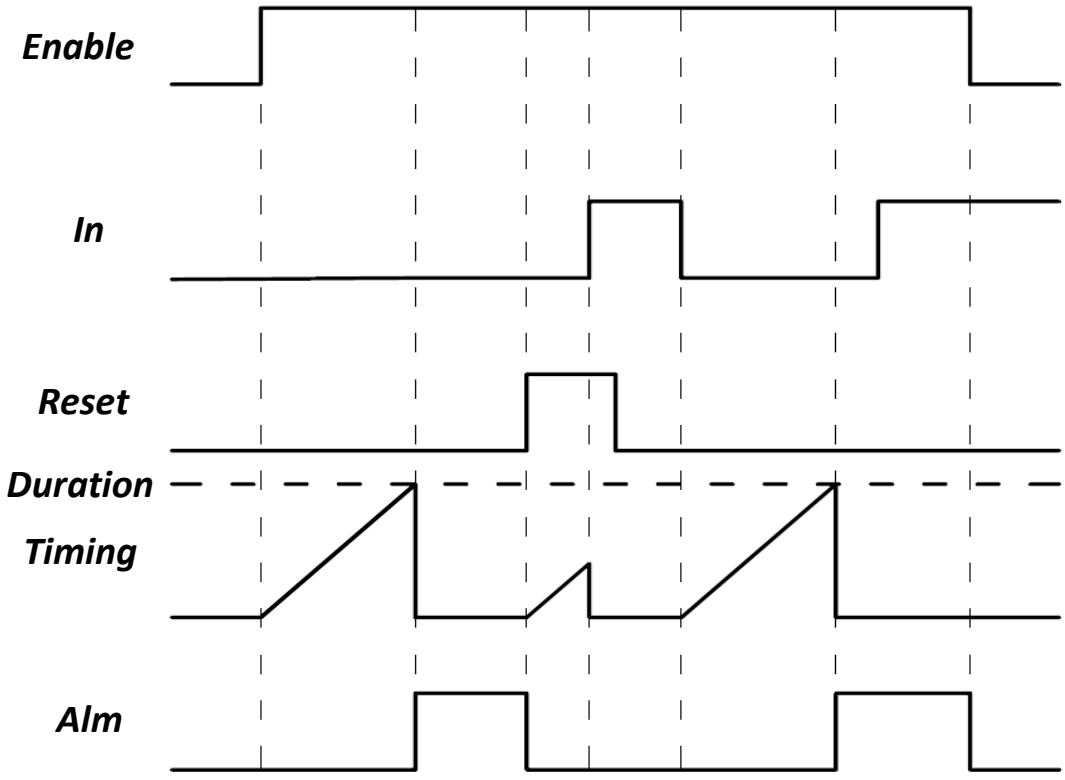
DAlarm



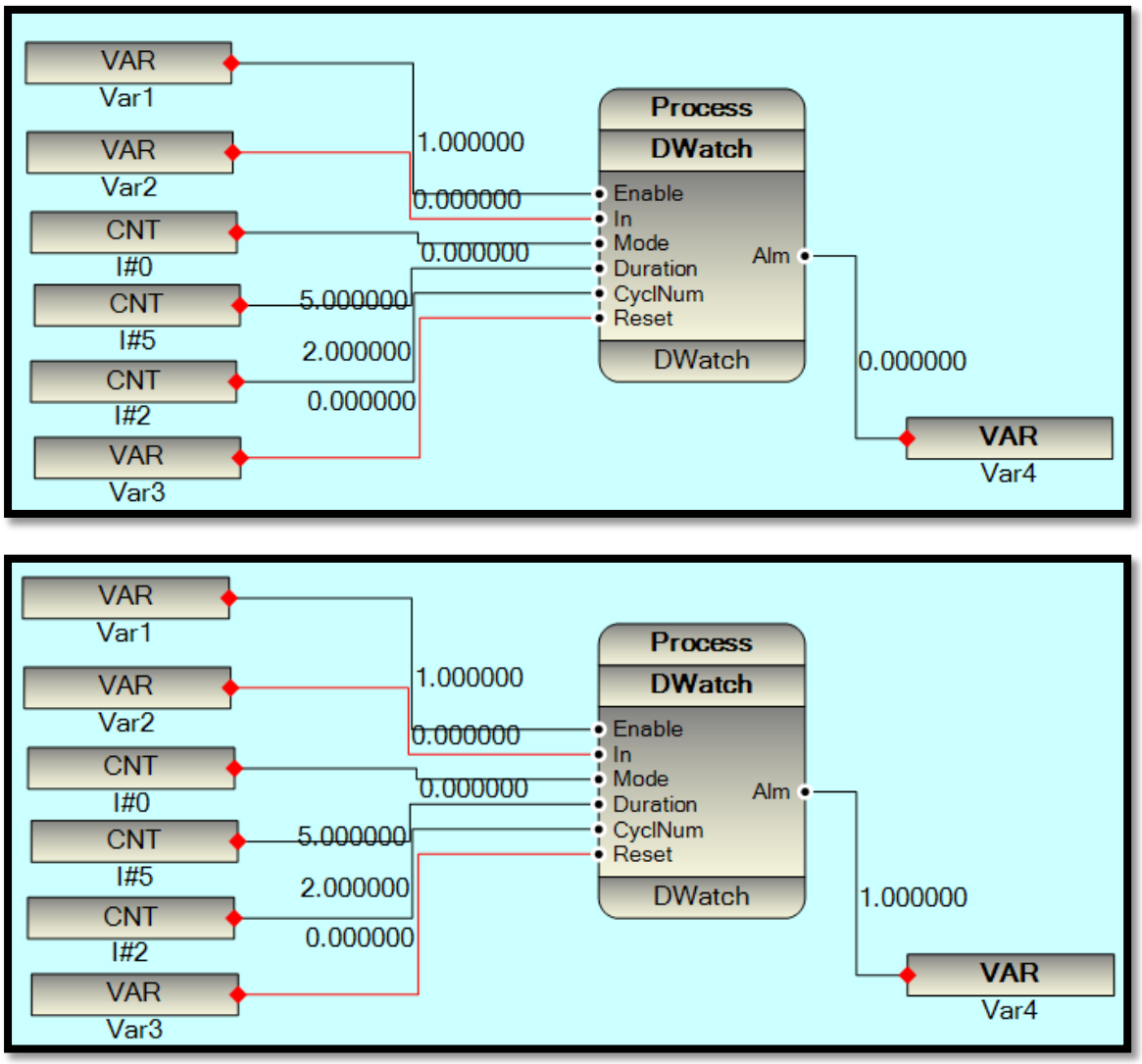
I/O	Data Type	Description
Enable	Double	*Enable/Disable*
In	Double	*Trigger*
Mode	Double	*Selecting Mode*
Duration	Double	*Duration*
CyclNum	Double	*Cycling number*
Reset	Double	*Reset*
Alm	Double	*Alarm*

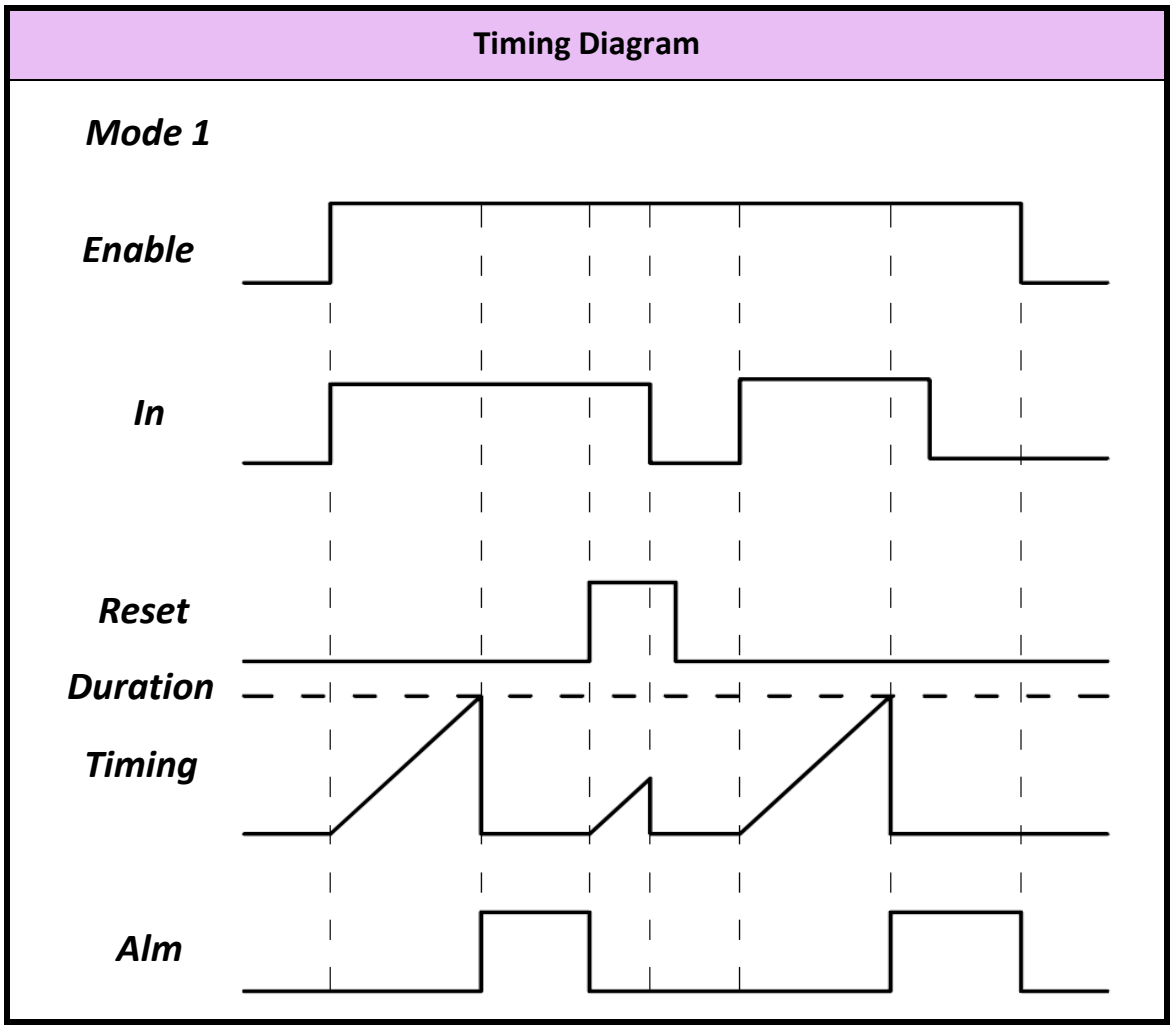
Timing Diagram

Mode 0

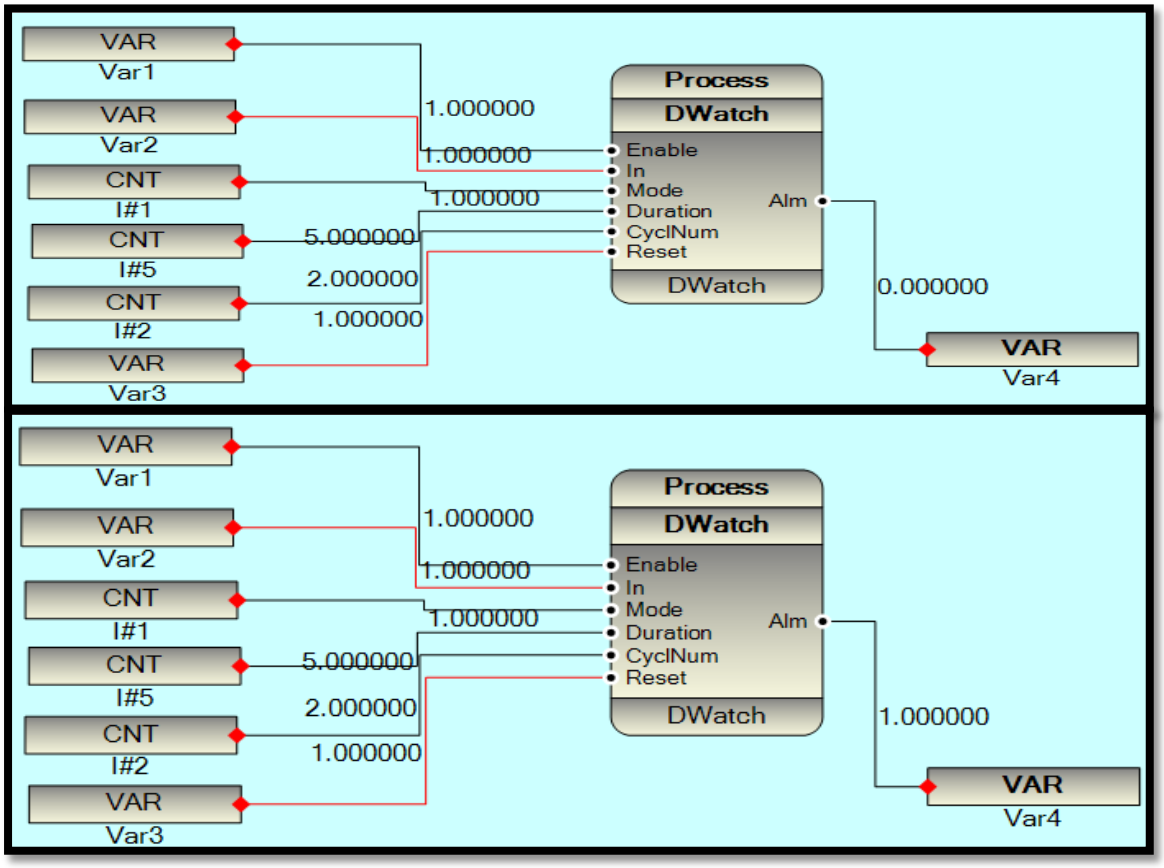


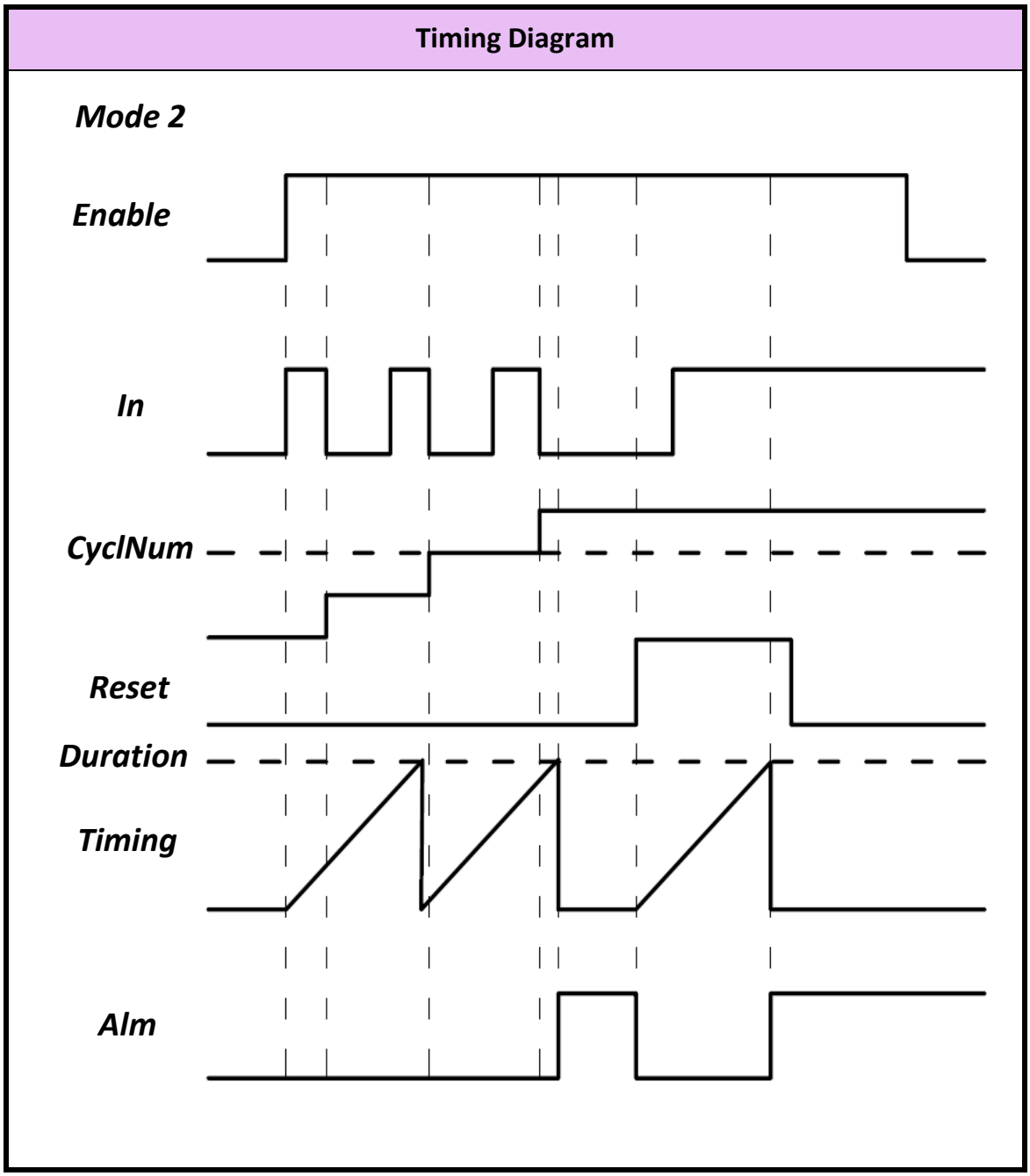
For Example: Mode 0



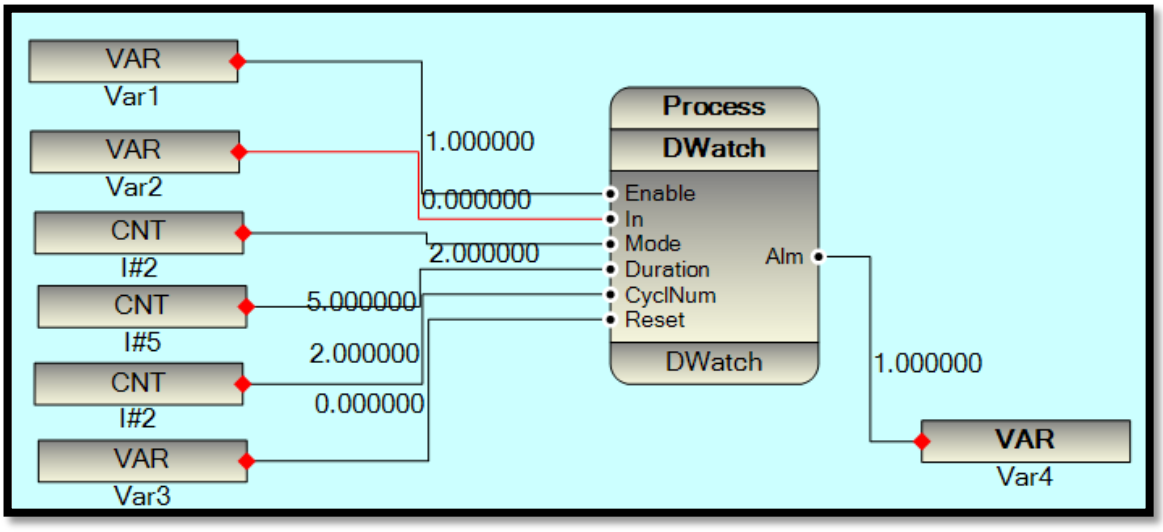


For Example: Mode 1





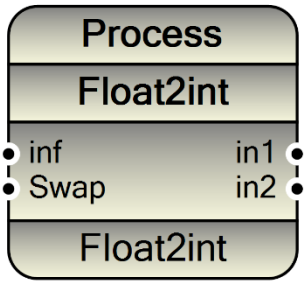
For Example: Mode 2

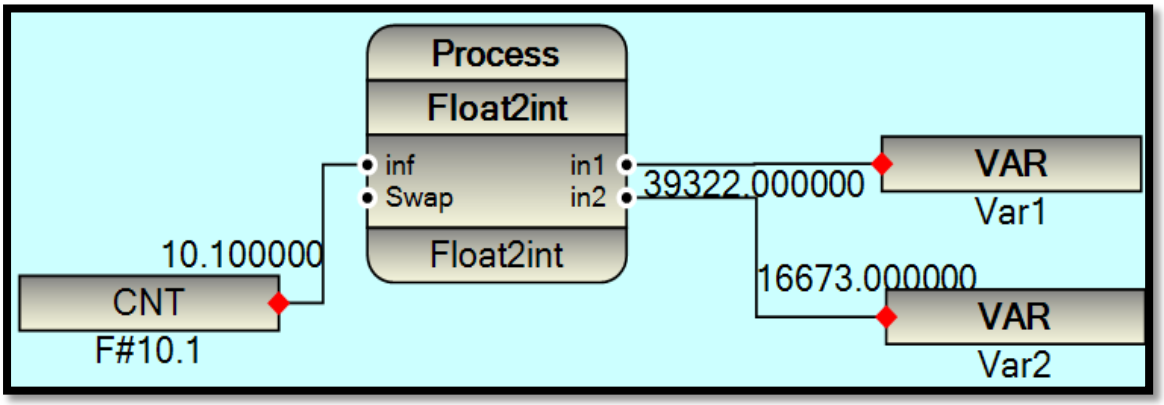


21.5.14 RunHours Function Block

21.5.15 Float2int Function Block

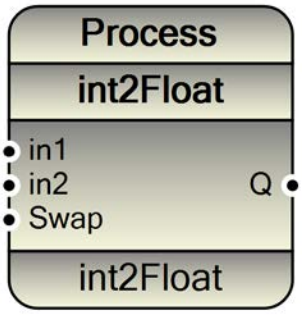
This Function Block is converter Float numbers to Integer numbers.

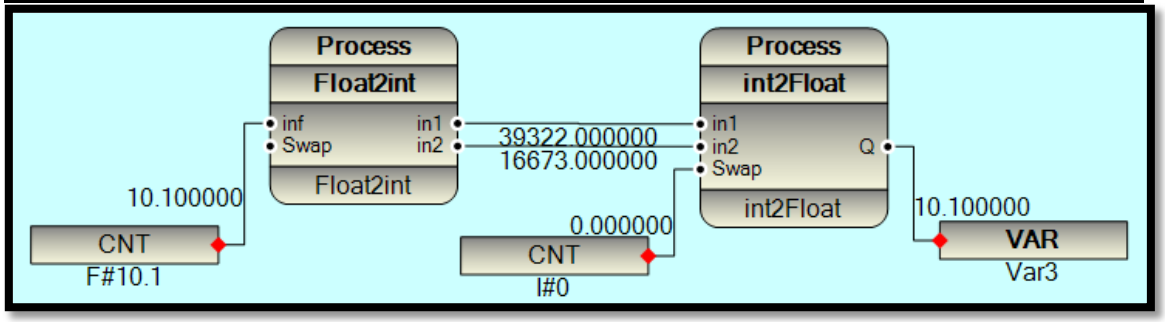
Float2int			
	**I/O**	**Data Type**	**Description**
	inf	**Double**	*Floating number*
	Swap	**Double**	*NC*
	in1	**Double**	*Integer number(16bit)*
	In2	**Double**	*Integer number(16bit)*



21.5.16 int2Float Function Block

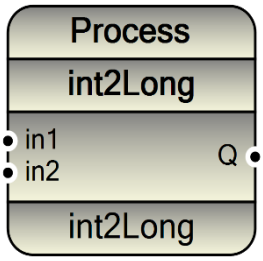
This Function Block is converter Integer numbers to Float numbers.

int2Float			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*Integer number(16bit)*
	In2	**Double**	*Integer number(16bit)*
	Swap	**Double**	*When Swap is 1,input number is swapped*
	Q	**Double**	*Floating number*

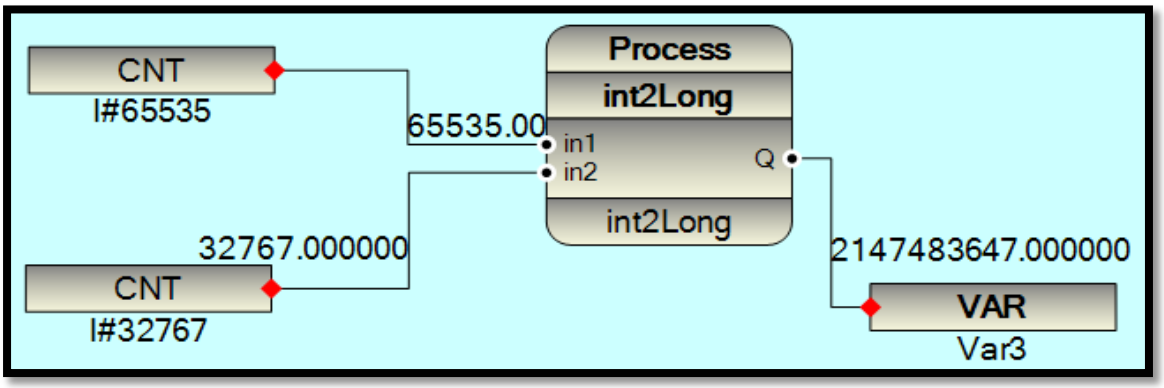


21.5.17 int2Long Function Block

This Function Block is converter Integer numbers to Long numbers.

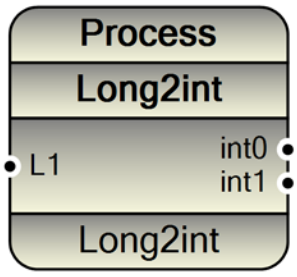
int2Long			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*Low value Integer number(16bit)*
	In2	**Double**	*High value Integer number(16bit)*
	Q	**Double**	*Floating number*

For Example: (0x 7FFF . 0x FFFF)=0x7FFFFFFF

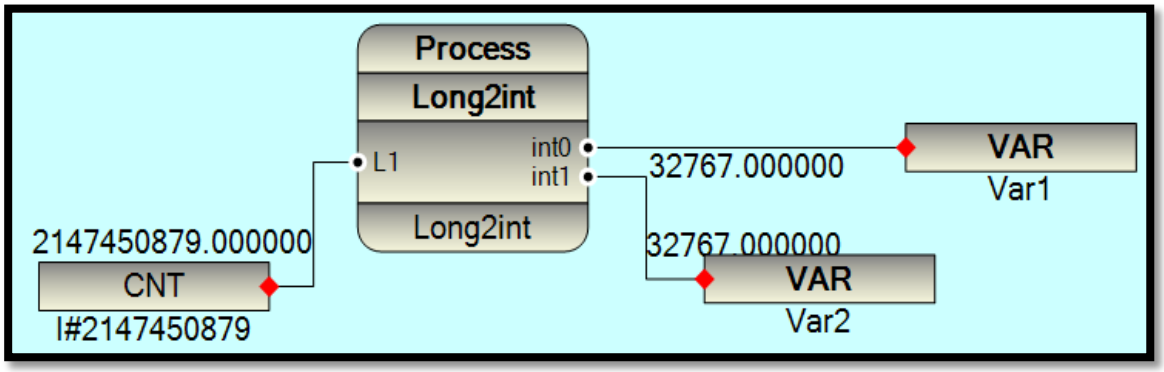


21.5.18 Long2int Function Block

This Function Block is converter Long numbers to Integer numbers.

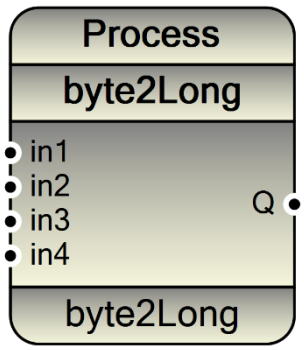
Long2int			
	**I/O**	**Data Type**	**Description**
	L1	**Double**	*Long numbers(32bit)*
	In0	**Double**	*Low value Integer number(16bit)*
	In1	**Double**	*High value Integer number(16bit)*

For Example: 0x7FFF7FFF=(0x 7FFF . 0x 7FFF)

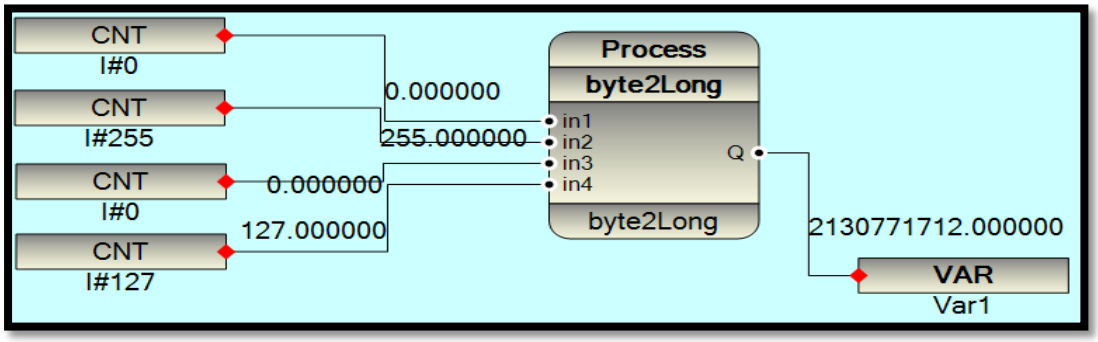


21.5.19 Byte2Long Function Block

This Function Block is converter byte numbers to Long numbers.

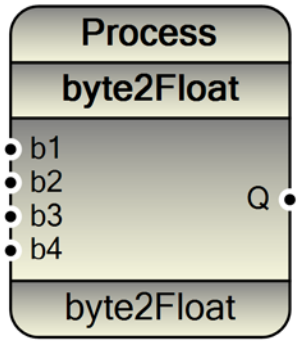
Byte2Long			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*(least valuable)Byte0*
	in2	**Double**	*Byte1*
	in3	**Double**	*Byte2*
	in4	**Double**	*(Most valuable)Byte3*
	Q	**Double**	*Long numbers(32bit)*

For Example: $(0x7F, 0x00, 0xFF, 0x00) = 0x7F00FF00$

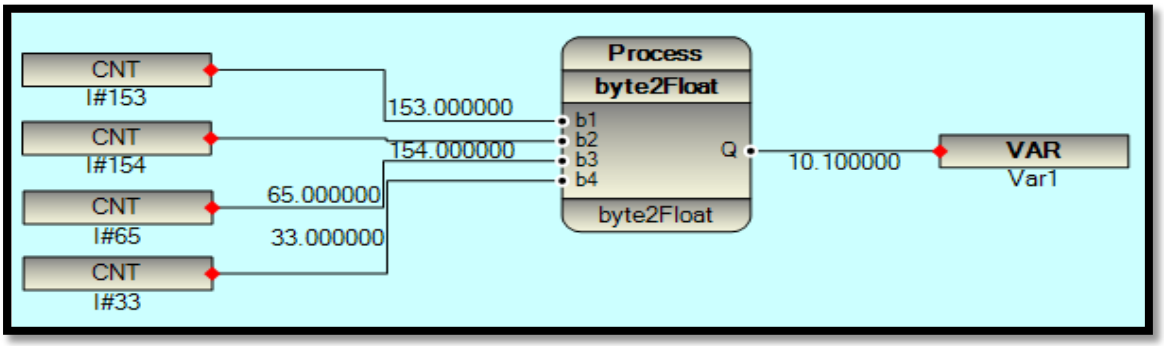


21.5.20 Byte2Float Function Block

This Function Block is converter byte numbers to Float numbers.

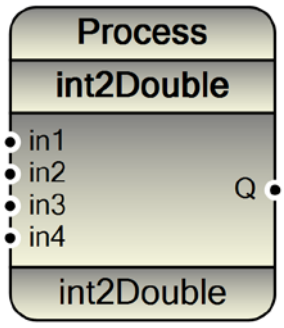
Byte2Float			
	**I/O**	**Data Type**	**Description**
	b1	**Double**	*(least valuable)Byte0*
	b2	**Double**	*Byte1*
	b3	**Double**	*Byte2*
	b4	**Double**	*(Most valuable)Byte3*
	Q	**Double**	*Float number*

For Example:

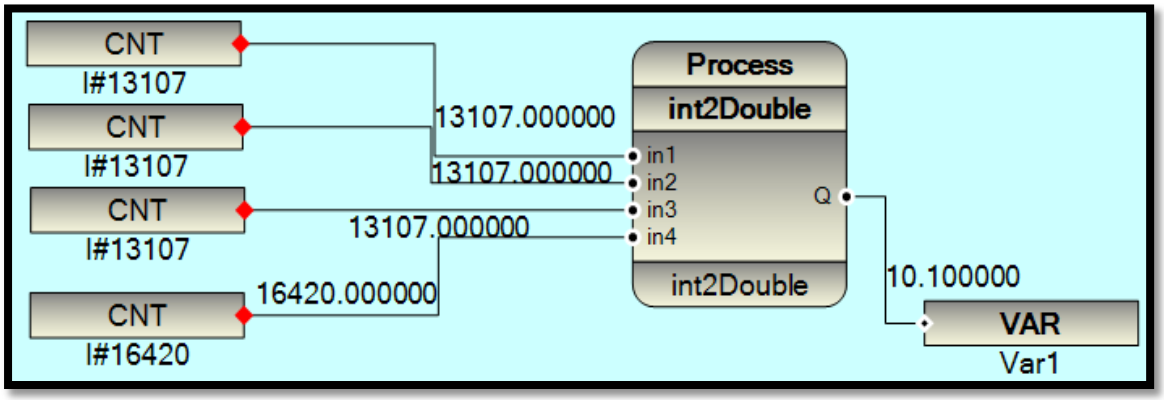


21.5.21 int2Double Function Block

This Function Block is converter Integer to Double.

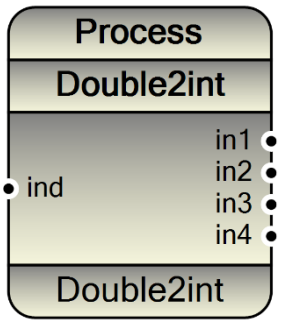
int2Double			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*Word0(16bit)*
	in2	**Double**	*Word 1*
	in3	**Double**	*Word 2*
	in4	**Double**	*Word 3*
	Q	**Double**	*Double(32bit)*

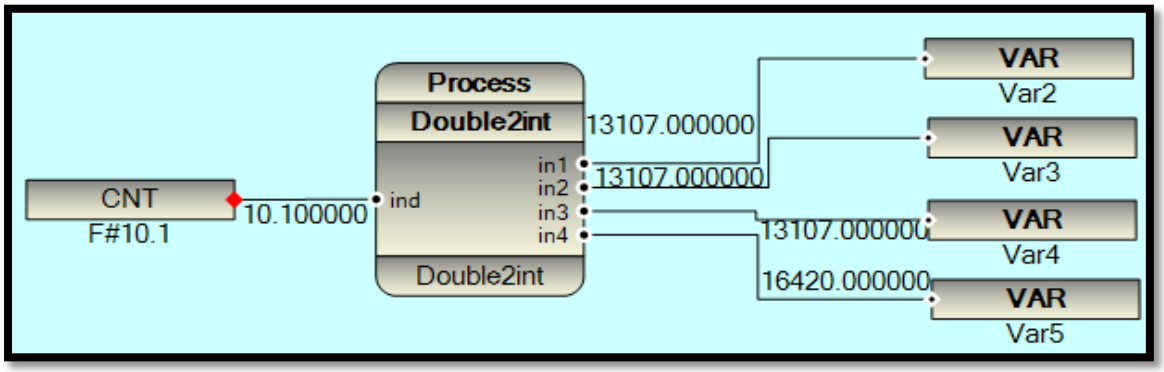
For Example:



21.5.22 Double2int Function Block

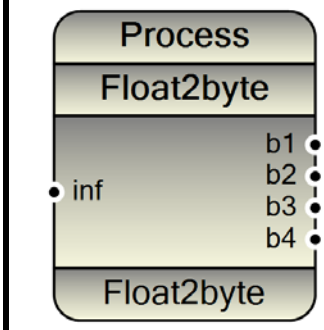
This Function Block is converter Double to Integer.

Double2int			
	**I/O**	**Data Type**	**Description**
	ind	**Double**	*Double(32bit)*
	in1	**Double**	*Word0(16bit)*
	in2	**Double**	*Word 1*
	in3	**Double**	*Word 2*
	in4	**Double**	*Word 3*

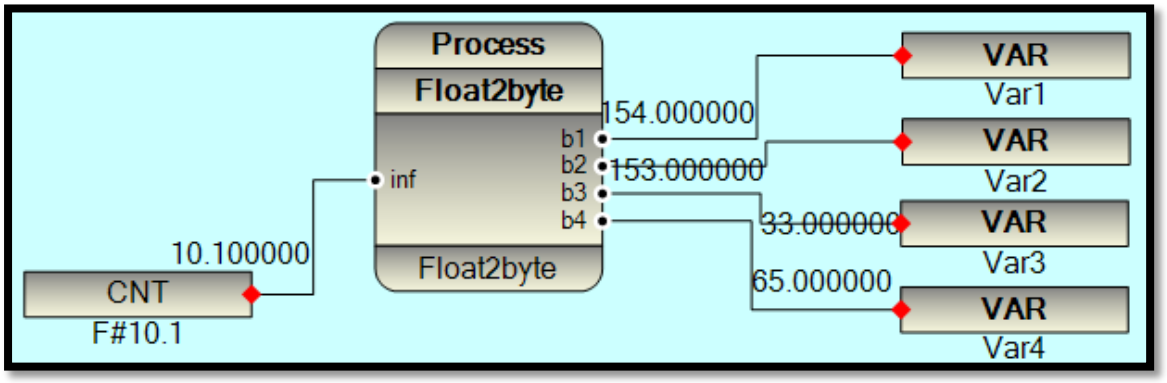


21.5.23 Float2byte Function Block

This Function Block is converter Float to byte.

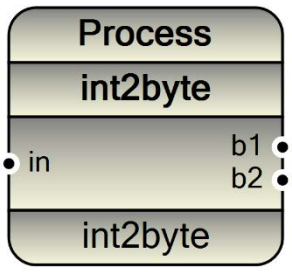
Float2byte			
	**I/O**	**Data Type**	**Description**
	inf	**Double**	*Float(16bit)*
	b1	**Double**	*(least valuable)Byte0*
	b2	**Double**	*Byte1*
	b3	**Double**	*Byte2*
	b4	**Double**	*(Most valuable)Byte3*

For Example:

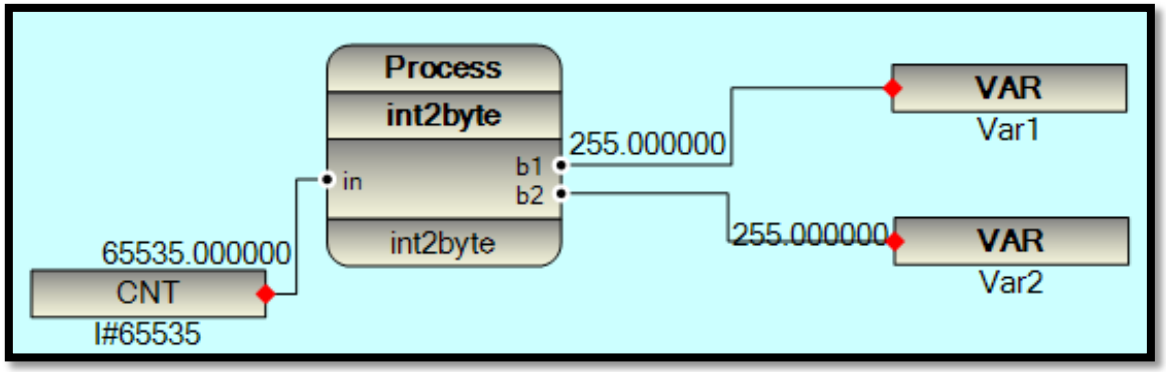


21.5.24 int2byte Function Block

This Function Block is converter Integer to byte.

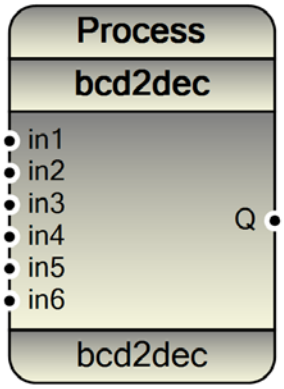
Float2byte			
	**I/O**	**Data Type**	**Description**
	inf	**Double**	*Integer(16bit)*
	b1	**Double**	*(least valuable)Byte0*
	b2	**Double**	*(Most valuable)Byte1*

For Example:

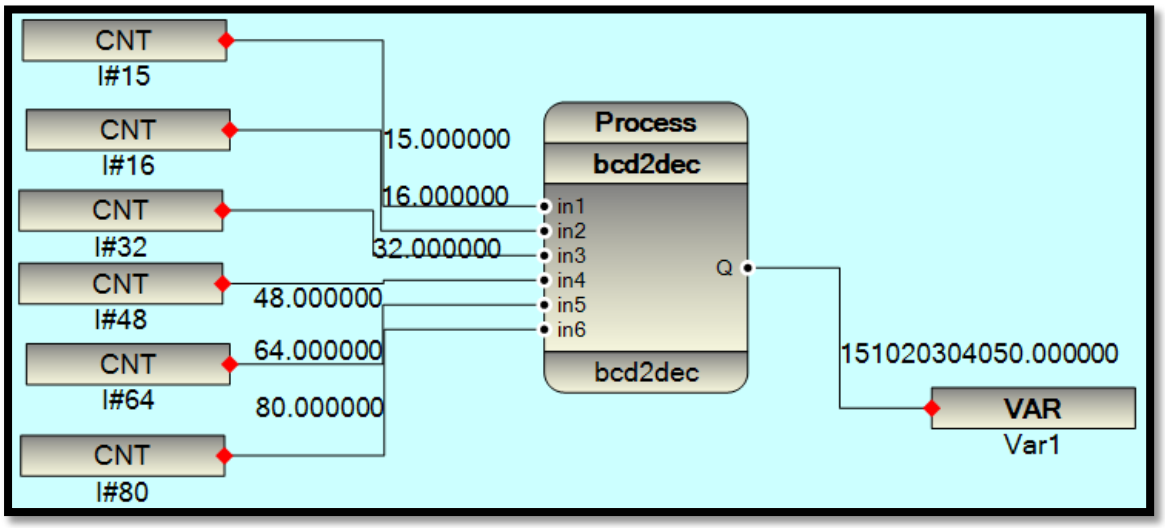


21.5.25 bcd2dec Function Block

This Function Block is converter bcd to decimal.

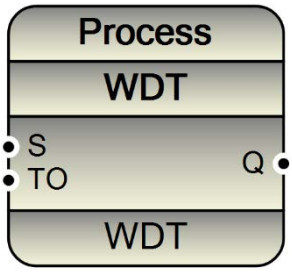
bcd2dec			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*Integer(16bit)*
	In2	**Double**	*Integer number*
	In3	**Double**	*Integer number*
	In4	**Double**	*Integer number*
	In5	**Double**	*Integer number*
	In6	**Double**	*Integer number*
	Q	**Double**	*BCD number*

For Example: *(0F,10,20,30,40,50) = (151020304050)*

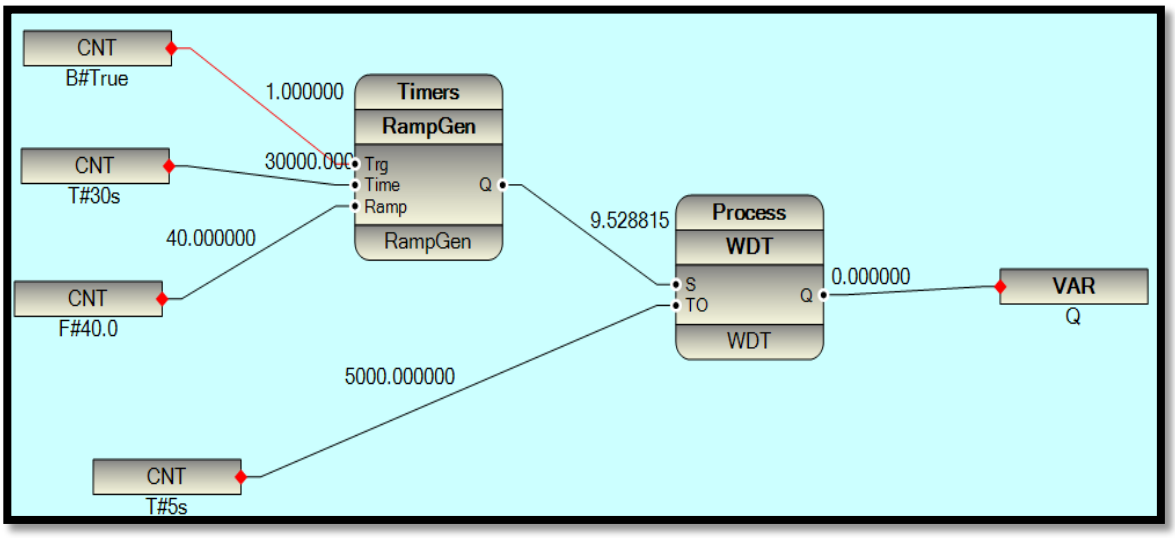


21.5.26 WDT Function Block

This Function is Watch Dog Function. If S input has any changes in less than TO, then Q Output is set to 0. If S Input is not changed for more than TO time, then Q Output will set to 1.

WDT			
	**I/O**	**Data Type**	**Description**
	S	**Double**	*Input Signal*
	TO	**Double**	*Watch Dog Time*
	Q	**Double**	*Output*

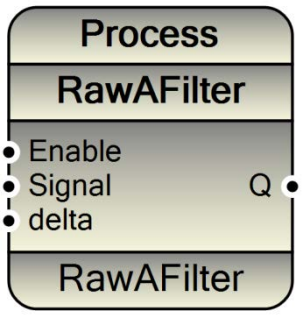
For Example:



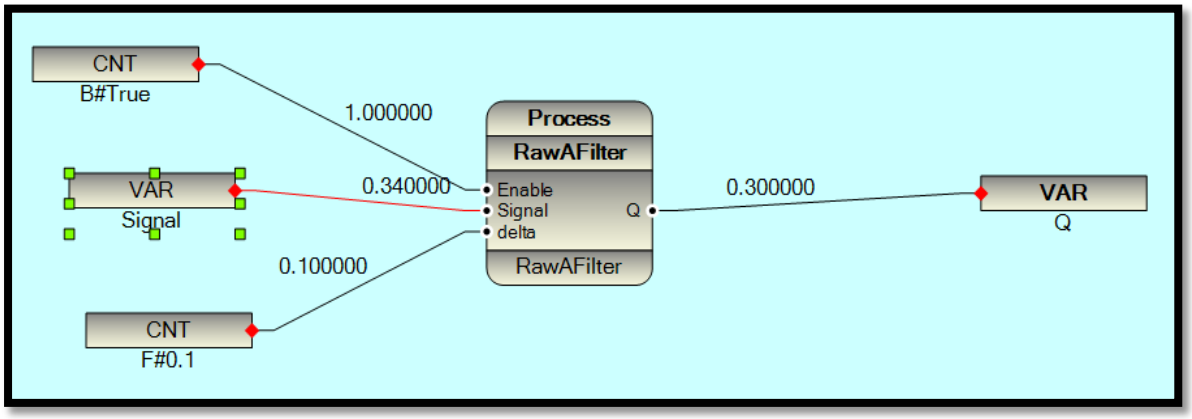
21.5.27 Scheduler Function Block

21.5.28 RawAFilter Function Block

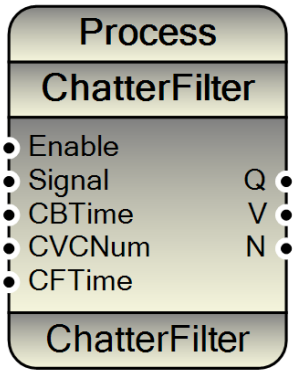
This Function Block is Standard Digital Filter but it doesn't need Input Signal Min and Max range. If Signal Changes is more than delta, it will map to Output, otherwise old value will map. In abut example if Input Signal is changing for bigger than 0.1, Current value of S Input Signal will map to Output. Otherwise old signal will pass.

RawAFilter			
	**I/O**	**Data Type**	**Description**
	Enabel	**Double**	*When Set to 1 , FB is operational*
	Signal	**Double**	*Input Signal*
	delta	**Double**	*When Input Signal Changes is more than Delta (in percentage) then Current value of Signal will pass to Q Output .*
	Q	**Double**	*Output*

For Example:

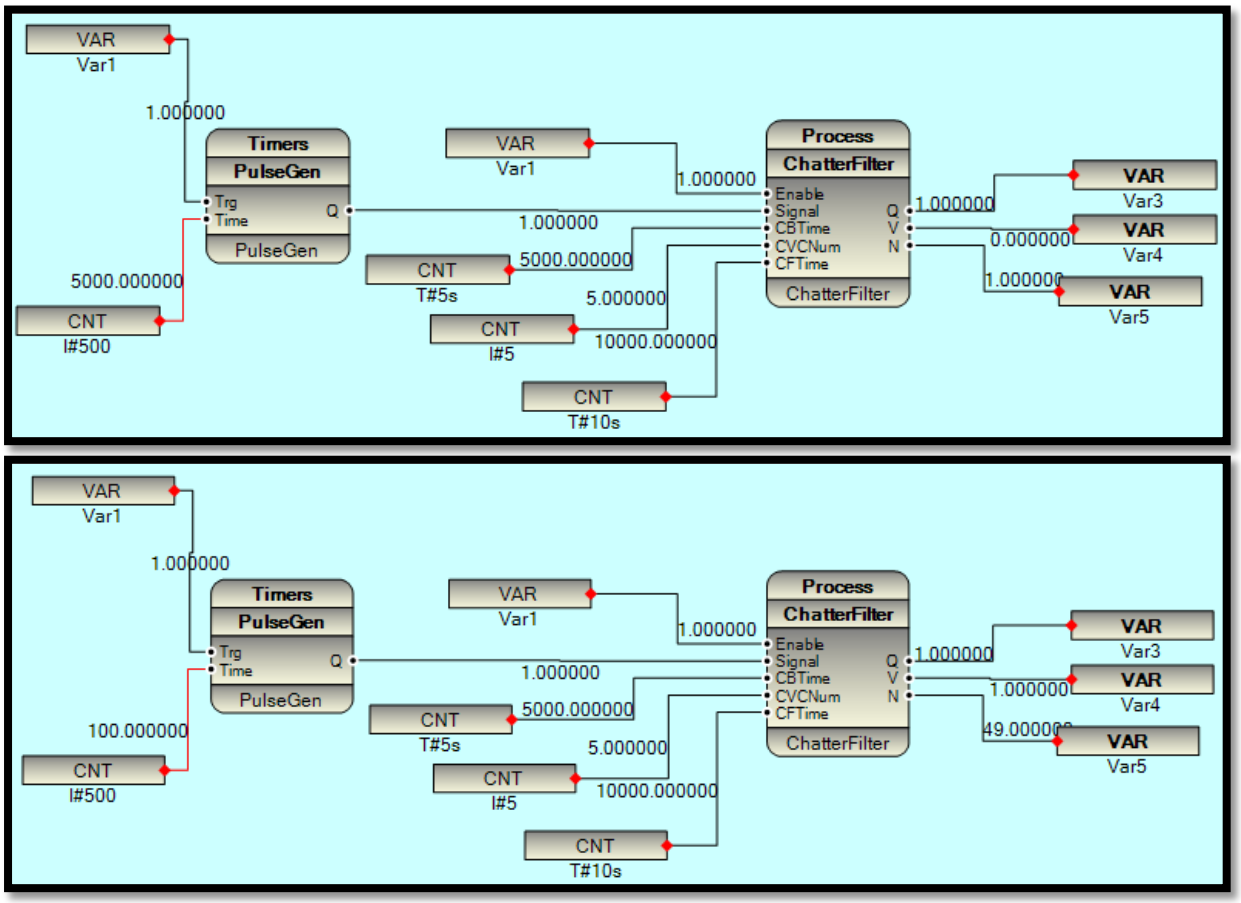


21.5.29 ChatterFilter Function Block



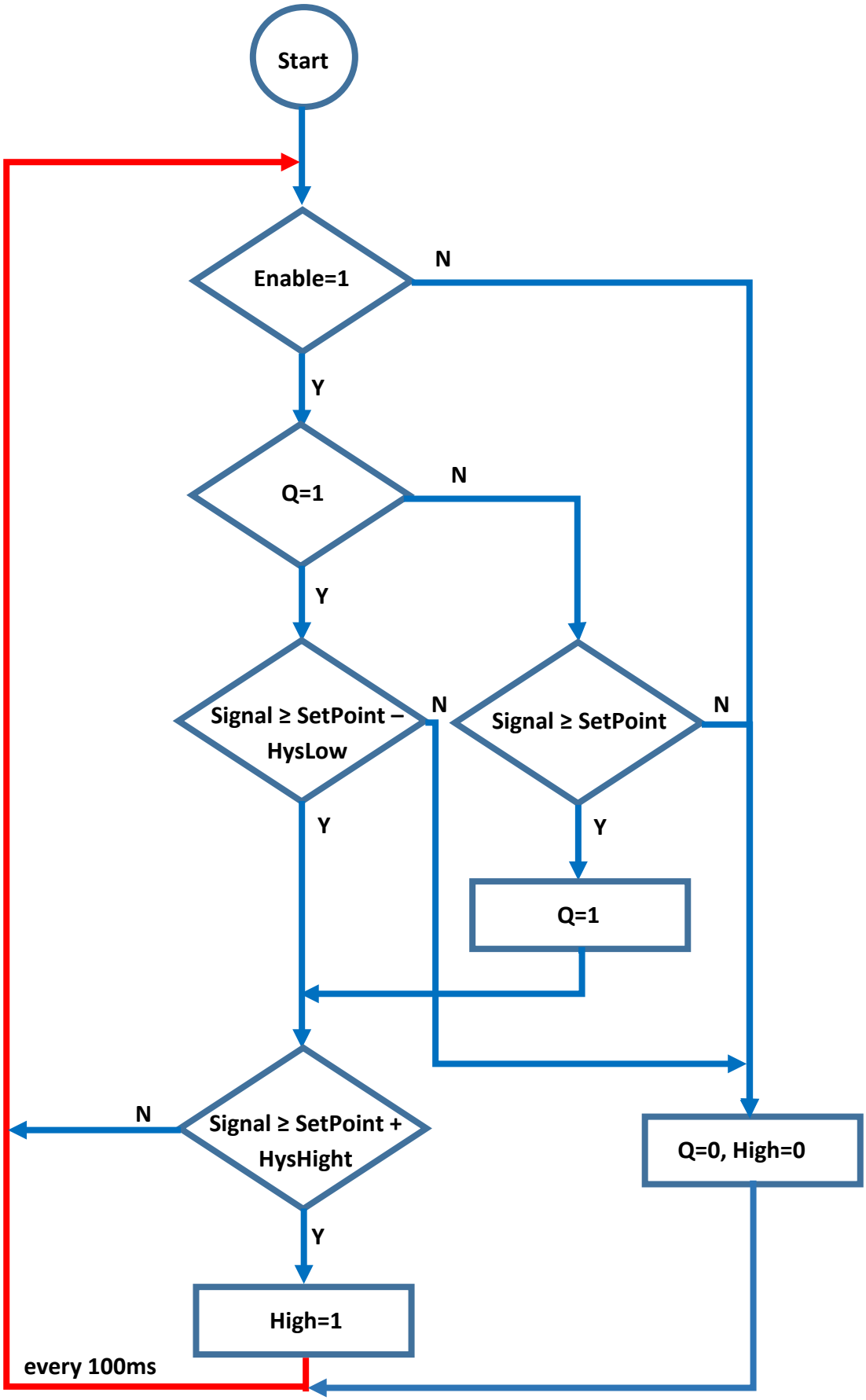
RawAFilter		
I/O	**Data Type**	**Description**
Enabel	**Double**	*Enable/Disable*
Signal	**Double**	*Signal Input*
CBTime	**Double**	*sampling time*
CVCNum	**Double**	*When* $N > CVCNum \rightarrow$ *Output is freezed*
CFTime	**Double**	*Freezing time*
Q	**Double**	*output*
V	**Double**	*When Output is freezed* $\rightarrow V = 1$
N	**Double**	*Number of changes*

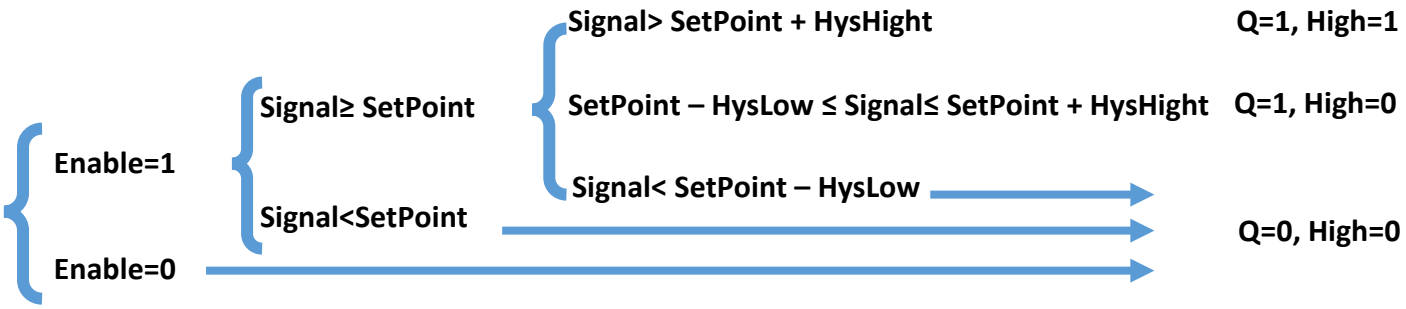
For Example:

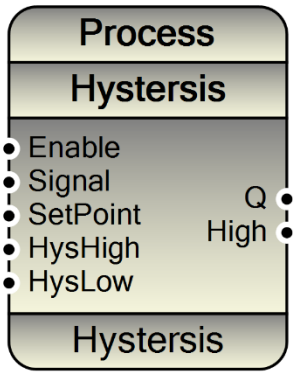

VAR
Var1
1.000000
Timers
PulseGen
Trg
Time
Q
PulseGen
5000.000000
CNT
I#500
VAR
Var1
1.000000
CNT
T#5s
5000.000000
CNT
I#5
5.000000
10000.000000
CNT
T#10s
Process
ChatterFilter
Enable
Signal
CBTime
CVCNum
CFTime
Q
V
N
ChatterFilter
1.000000
0.000000
1.000000
VAR
Var3
VAR
Var4
VAR
Var5
VAR
Var1
1.000000
Timers
PulseGen
Trg
Time
Q
PulseGen
100.000000
CNT
I#500
VAR
Var1
1.000000
CNT
T#5s
5000.000000
CNT
I#5
5.000000
10000.000000
CNT
T#10s
Process
ChatterFilter
Enable
Signal
CBTime
CVCNum
CFTime
Q
V
N
ChatterFilter
1.000000
1.000000
49.000000
VAR
Var3
VAR
Var4
VAR
Var5

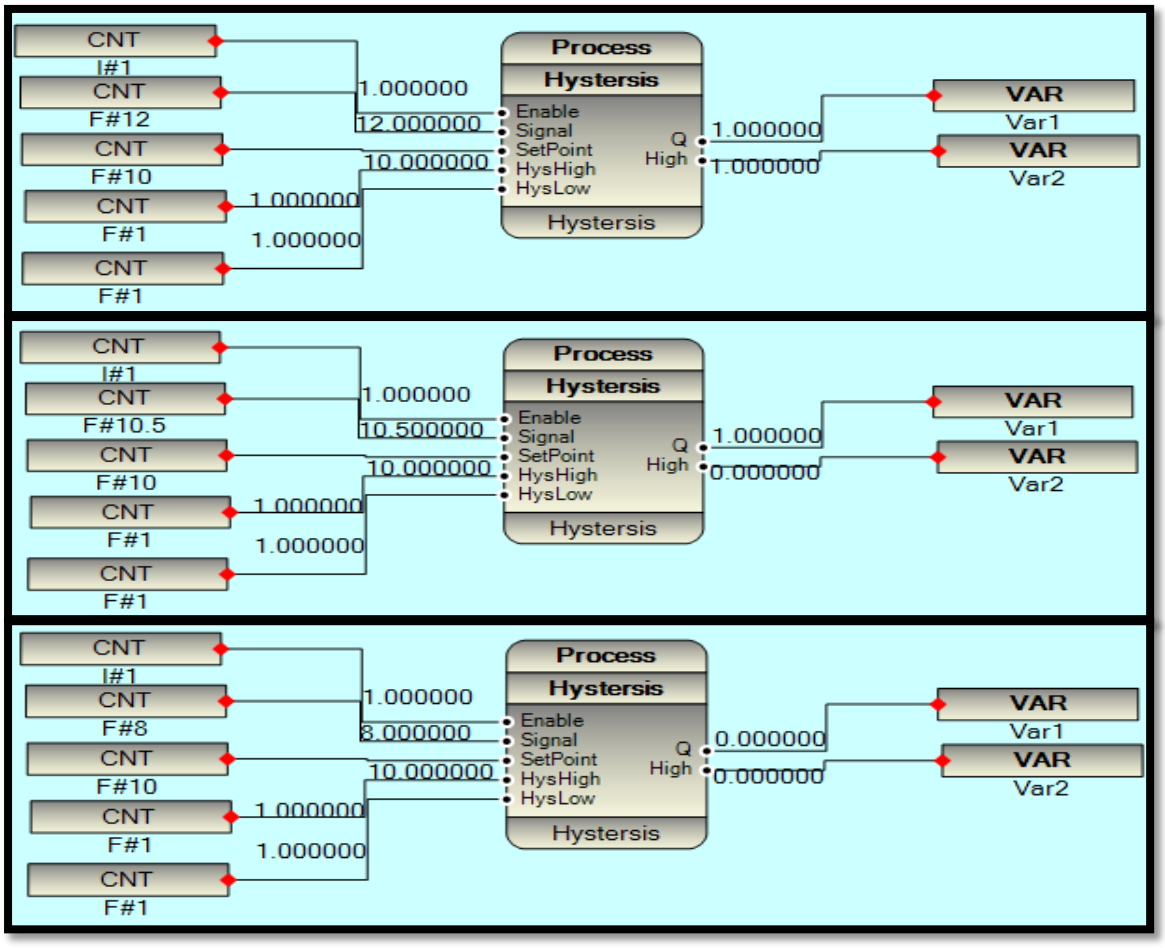
- **21.5.30 RawAFilterLP Function Block**
- **21.5.31 Hysteresis Function Block**

Hysteresis Function Block used for enable and disable the system in 2 level.





bcd2dec			
	**I/O**	**Data Type**	**Description**
	Enable	**Double**	*Enable/Disable*
	Signal	**Double**	*Input Signal*
	SetPoint	**Double**	*Set Point*
	HysHight	**Double**	*Hysteresis High*
	HysLow	**Double**	*Hysteresis Low*
	Q	**Double**	*Digital Output*
	High	**Double**	*High*



21.6 IEC1131-3 Group:

All Function blocks in this group are based on IEC1131-3 Standard. You will find some of this Function Blocks in Other group with different names. Function Block Name, Input Output Pins and Function definition is completely based on IEC1131-3 Standard.

21.7 Scheduling Group:

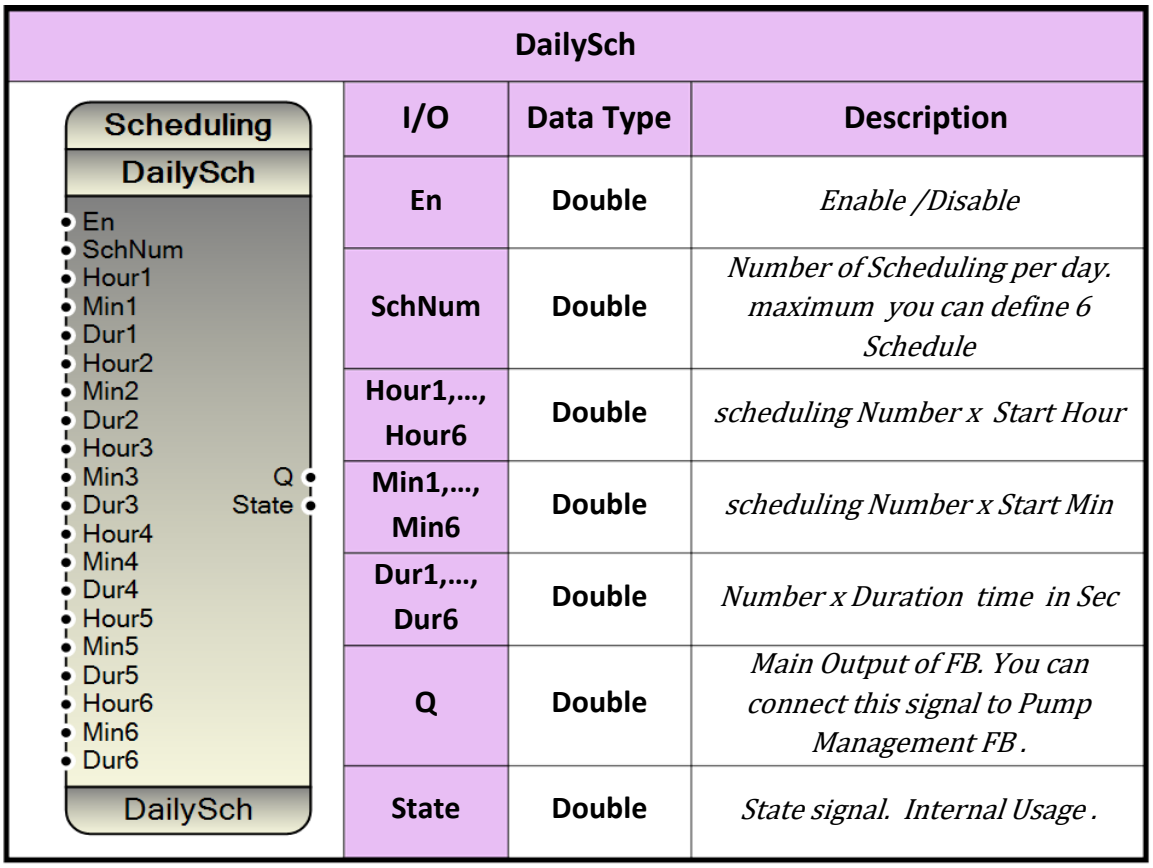
Process Function Groups

21.7.1 DailySch Function Block

These Function is used for daily scheduling of Output.

DailySch Function will check current Time in RTU and if it is found time for any Scheduling Time, it will make output High for Duration time.

Then you can use DailySch function with following parameters:

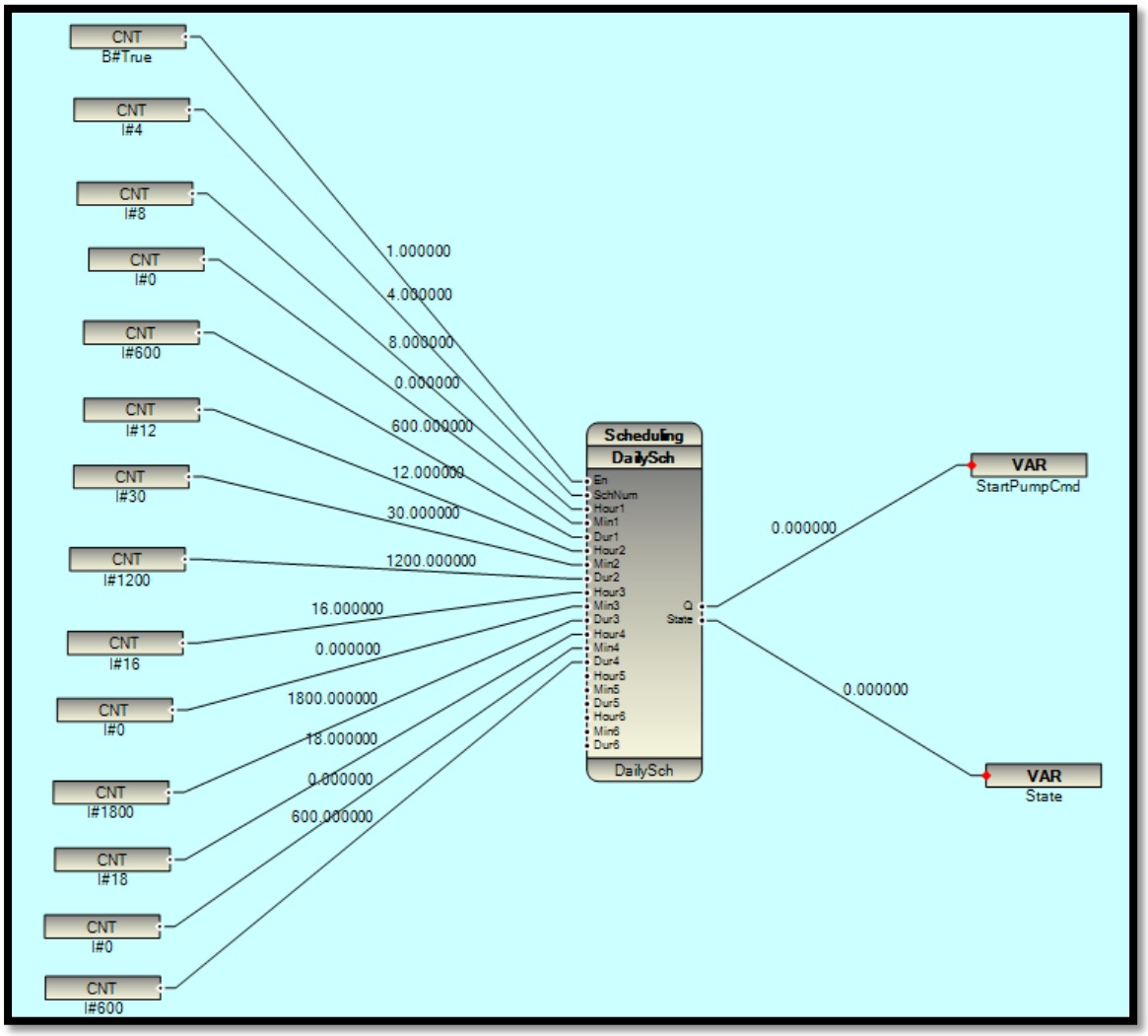


I/O	Data Type	Description
En	**Double**	*Enable /Disable*
SchNum	**Double**	*Number of Scheduling per day. maximum you can define 6 Schedule*
Hour1,..., Hour6	**Double**	*scheduling Number x Start Hour*
Min1,..., Min6	**Double**	*scheduling Number x Start Min*
Dur1,..., Dur6	**Double**	*Number x Duration time in Sec*
Q	**Double**	*Main Output of FB. You can connect this signal to Pump Management FB .*
State	**Double**	*State signal. Internal Usage .*

For Example:

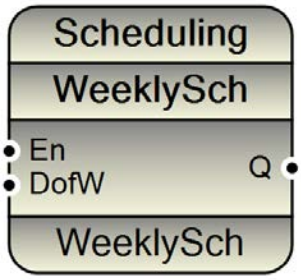
Suppose you want to start an irrigation pump daily with following scheduling:

08:00 start Pump for 10 Min
12:30 start Pump for 20 Min
16:00 Start Pump for 30 Min
18:00 Start Pump for 10 Min

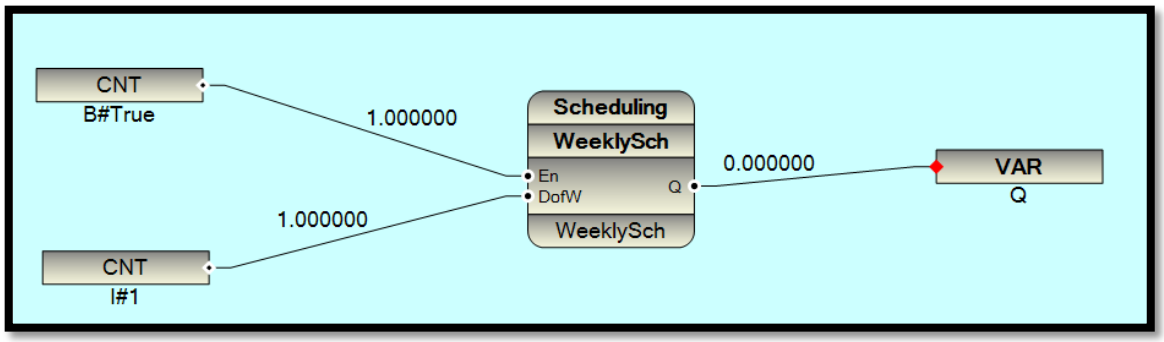


21.7.2 WeeklySch Function Block

These Function check current day of week and set Q if it is same as DofW (Day of Week). DofW is between 0 (Sunday) to 6.
You can use WeeklySch with DailySch to make different Scheduling for every day of week.

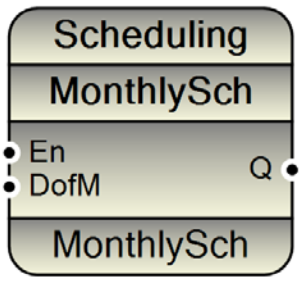
WeeklySch			
	**I/O**	**Data Type**	**Description**
	En	**Double**	*Enable /Disable*
	DofW	**Double**	*Day of Week .DofW is between 0 (Sunday) to 6*
	Q	**Double**	*Output*

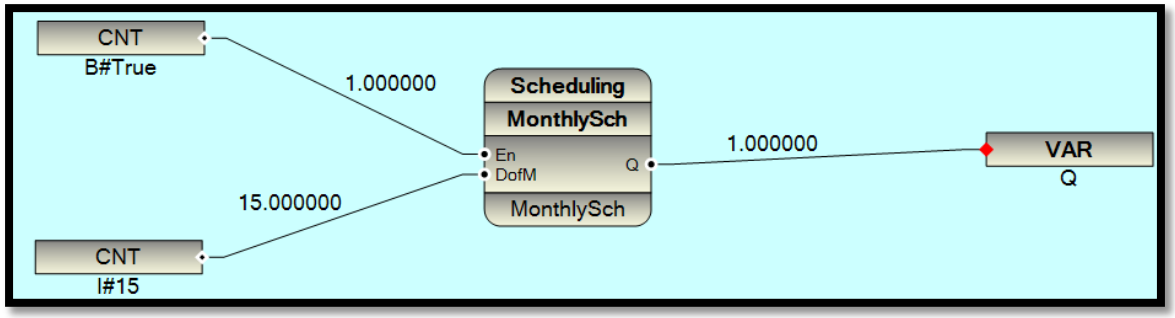
For Example:



21.7.3 MonthlySch Function Block

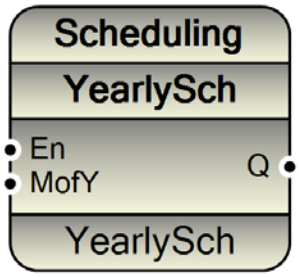
These Function check current day of month and set Q if it is same as DofM (Day of Month). DofM is integer number between 1 and 31.

MonthlySch			
	**I/O**	**Data Type**	**Description**
	En	**Double**	*Enable /Disable*
	DofM	**Double**	*Day of Month. DofM is integer number between 1 and 31*
	Q	**Double**	*Output*

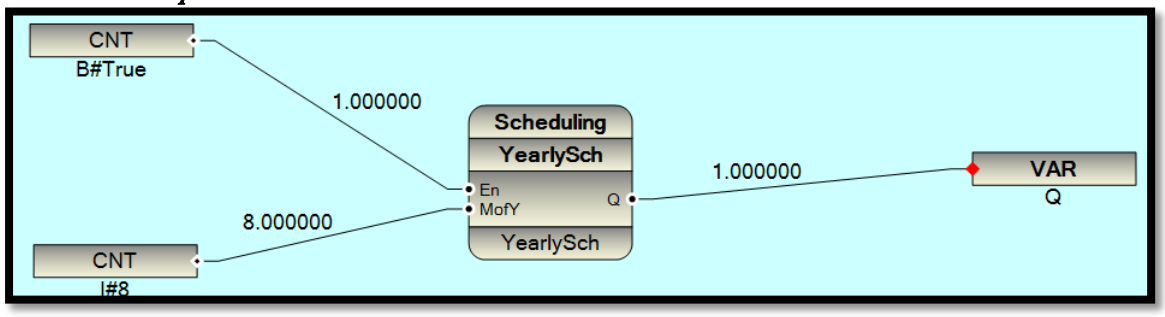


21.7.4 YearlySch Function Block

These Function check current Month with MofY (Month of Year), if it is same as MofY Q Output will set to high.
You can use YearlySch and MonthlySch with DailySch to make different scheduling based on different seasons.

YearlySch			
	**I/O**	**Data Type**	**Description**
	En	**Double**	*Enable /Disable*
	MofW	**Double**	*Month of Year. DofM is integer number between 1 and 12*
	Q	**Double**	*Output*

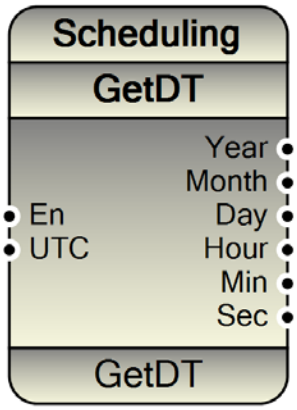
For Example:



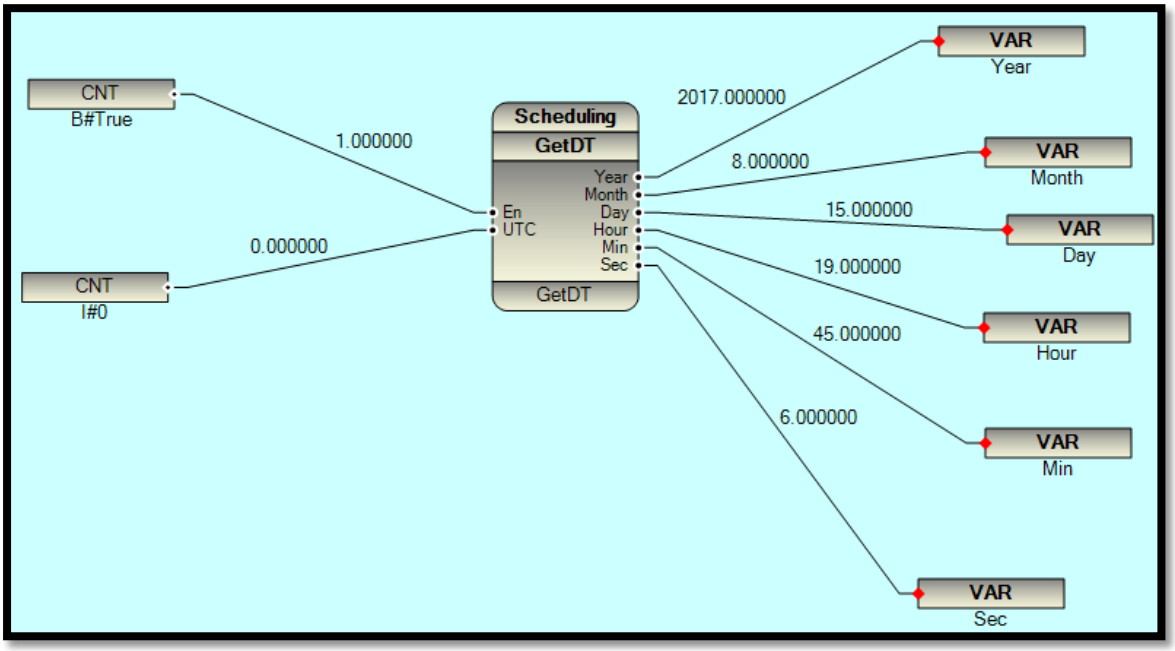
21.7.5 GetDT Function Block

These Function Shows Current Date and Time of RTU

If En = False, all Outputs are zero

GetDT			
	**I/O**	**Data Type**	**Description**
	En	**Double**	*Enable /Disable Function Block .*
	UTC	**Double**	*If UTC = 0, Output DT is in local time* *If UTC = 1 , Output is in UTC Time*
	Year	**Double**	*Year*
	Month	**Double**	*Month*
	Day	**Double**	*Day*
	Hour	**Double**	*Hour*
	Min	**Double**	*Minutes*
	Sec	**Double**	*Second*

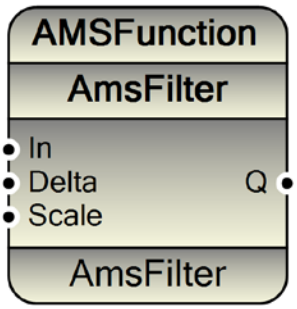
For Example:


CNT
B#True
1.000000
CNT
I#0
0.000000
Scheduling
GetDT
En
UTC
Year
Month
Day
Hour
Min
Sec
GetDT
2017.000000
8.000000
15.000000
19.000000
45.000000
6.000000
VAR
Year
VAR
Month
VAR
Day
VAR
Hour
VAR
Min
VAR
Sec

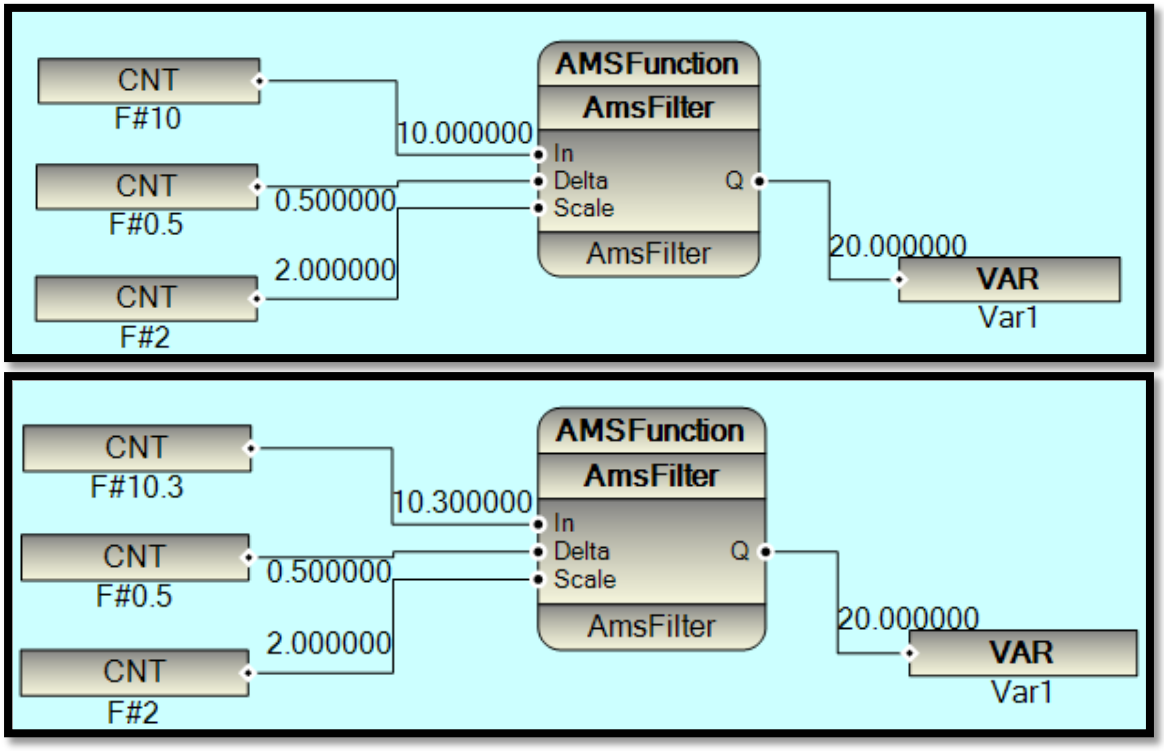
- **21.8 AMSFunctiom Group:**
  AMS Function Group
- **21.8.1 AmsDrive Function Block**
- **21.8.2 AmsDriveNP Function Block**
- **21.8.3 AmsFilter Function Block**

When Input Signal Changes is more than Delta then Current value of Signal multiplied in scale and will pass to Q Output.

In above example when Input signal changes is more than 0.5 then Input Signal multiplied in scale and will map to Output Signal, otherwise old value will pass to output.

AmsFilter			
	**I/O**	**Data Type**	**Description**
	In	**Double**	*Input Signal*
	Delta	**Double**	*Tolerance of error*
	Scale	**Double**	*Scale factor*
	Q	**Double**	*Output=Scale * In*

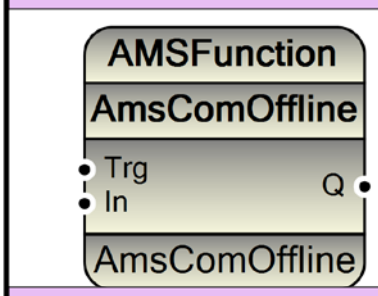
For Example:



21.8.4 AmsComOffline Function Block

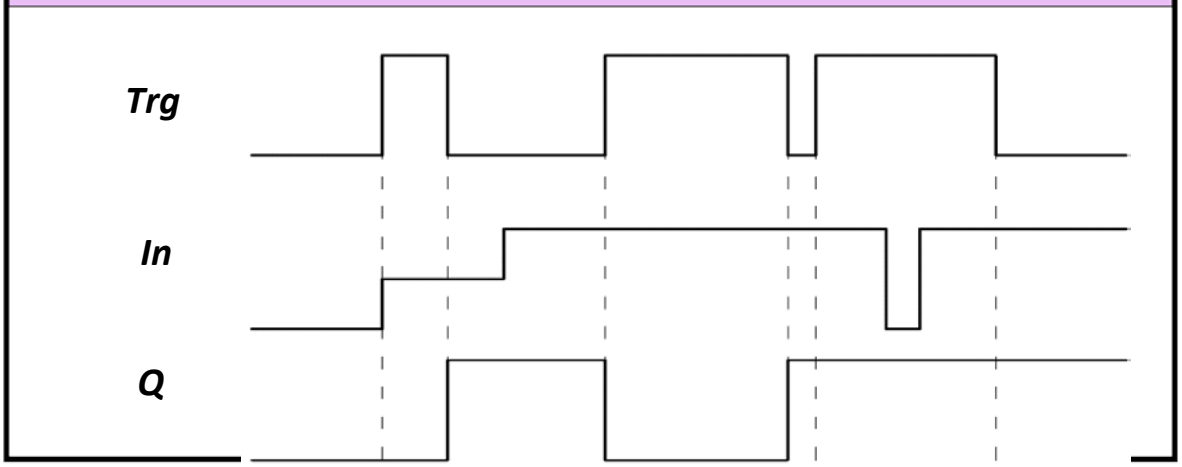
When Input Signal value changed from old signal value, Q is 0 but, when input signal value equal of old signal value, Q is 1.

AmsComOffline

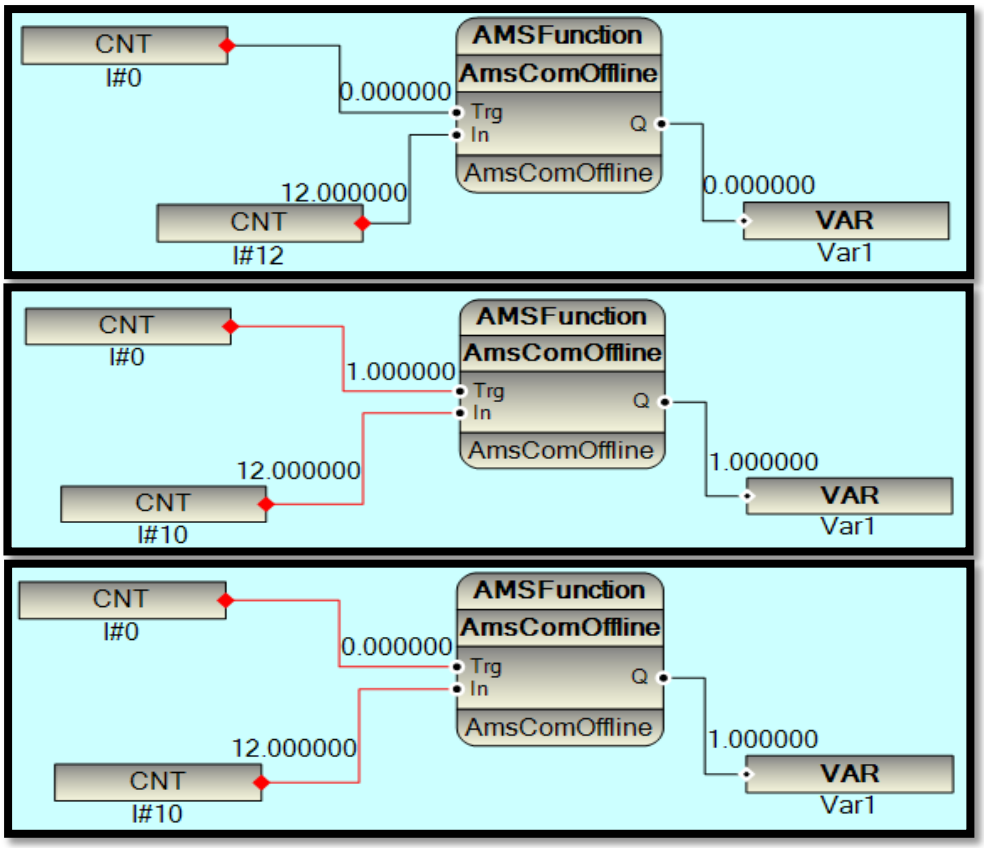


I/O	Data Type	Description
Trg	Double	*Trigger*
In	Double	*Input Value*
Q	Double	*Output*

Timing Diagram



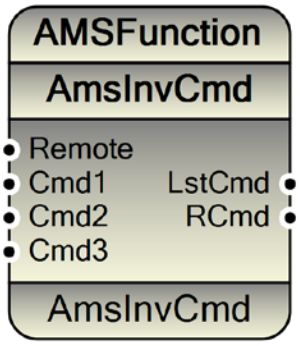
For Example:

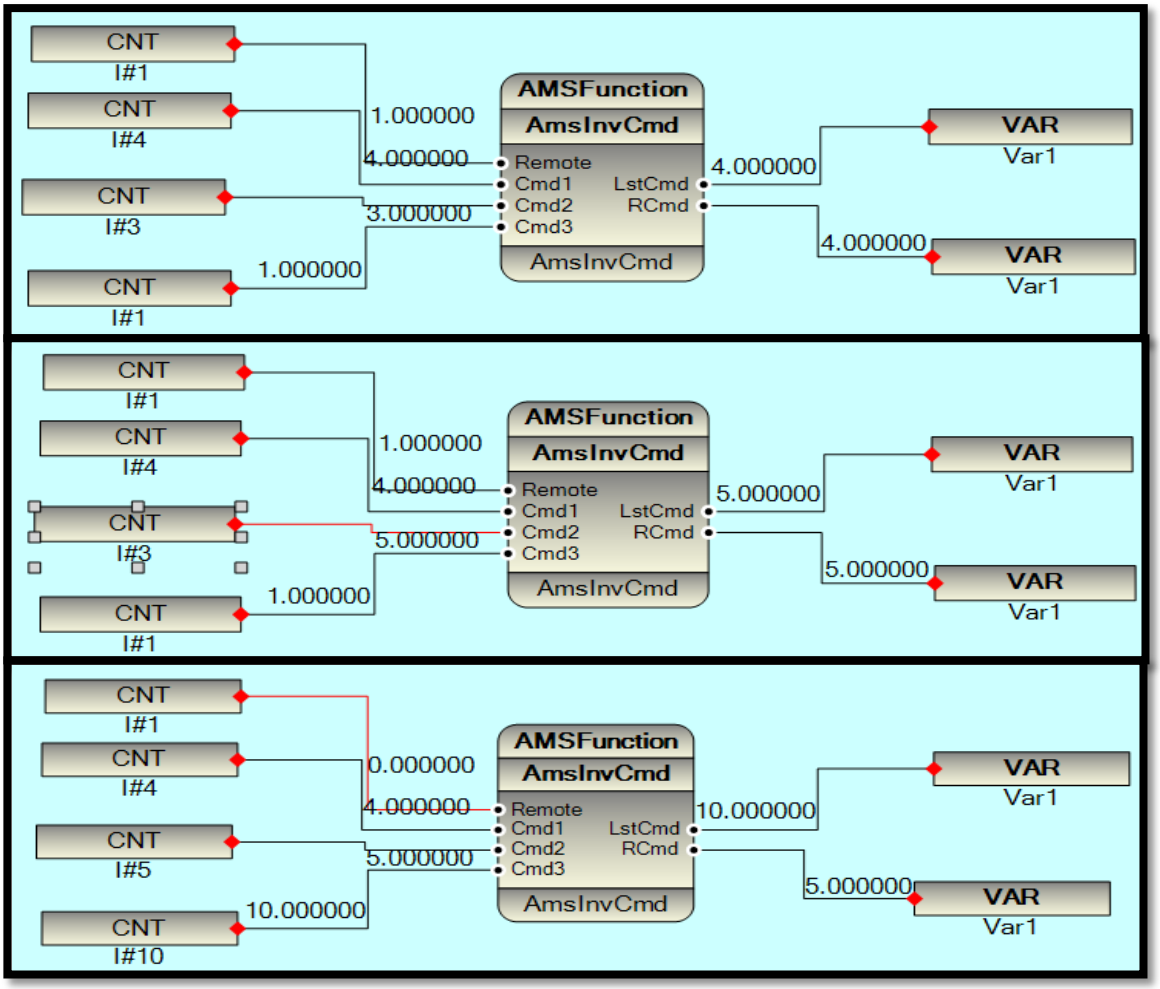


21.8.5 AmsInvCmd Function Block

When every one of cmd1 value or cmd2 value changed then old values RCmd is changed values.

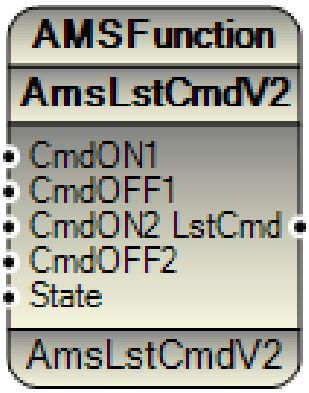
Remote=1: LstCmd=RCmd
Remote=0: LstCmd=Cmd3

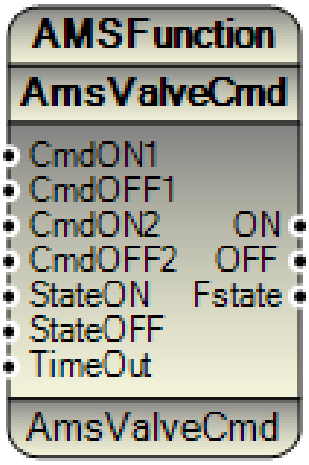
AmsInvCmd			
	**I/O**	**Data Type**	**Description**
	Remote	**Double**	*Remote*
	Cmd1	**Double**	*Input 1*
	Cmd2	**Double**	*Input2*
	Cmd3	**Double**	*Input3*
	LstCmd	**Double**	*When Remote is 0, LstCmd is Cmd3 When Remote is 1,LstCmd=RCmd*
	RCmd	**Double**	*RCmd select between Cmd1 or Cmd2*

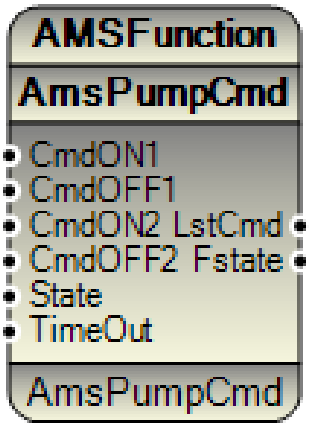


21.8.6 Cmd Function Block

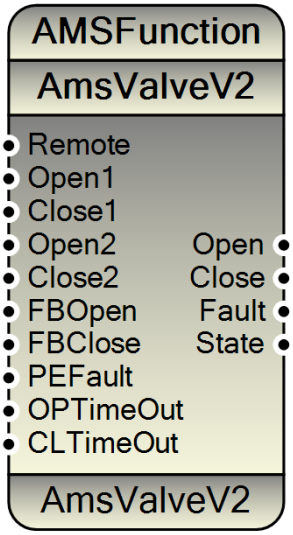
in This functions receive commands of systems and transmit final changes to output .for transmit data in function defined a timeout. when time passed of the timeout, the Output is disable.



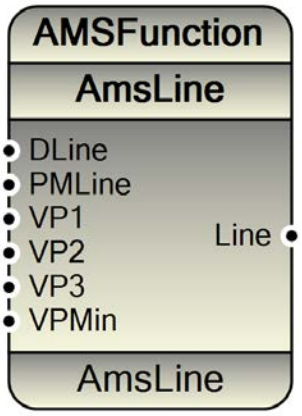




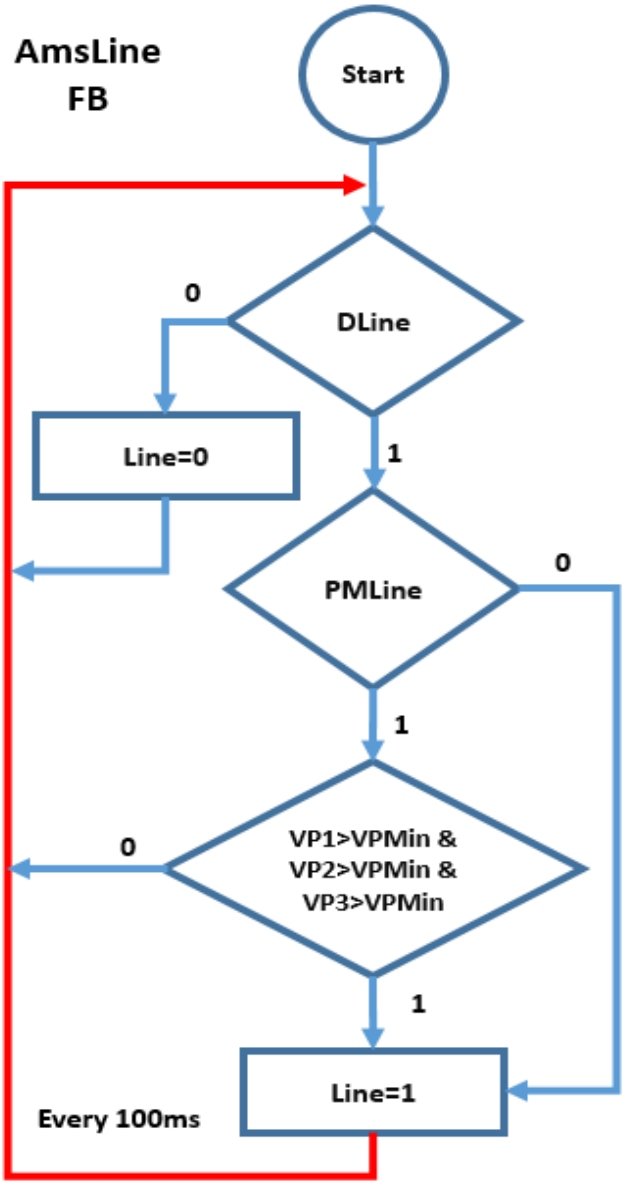
21.8.7 AmsValveV2 Function Block

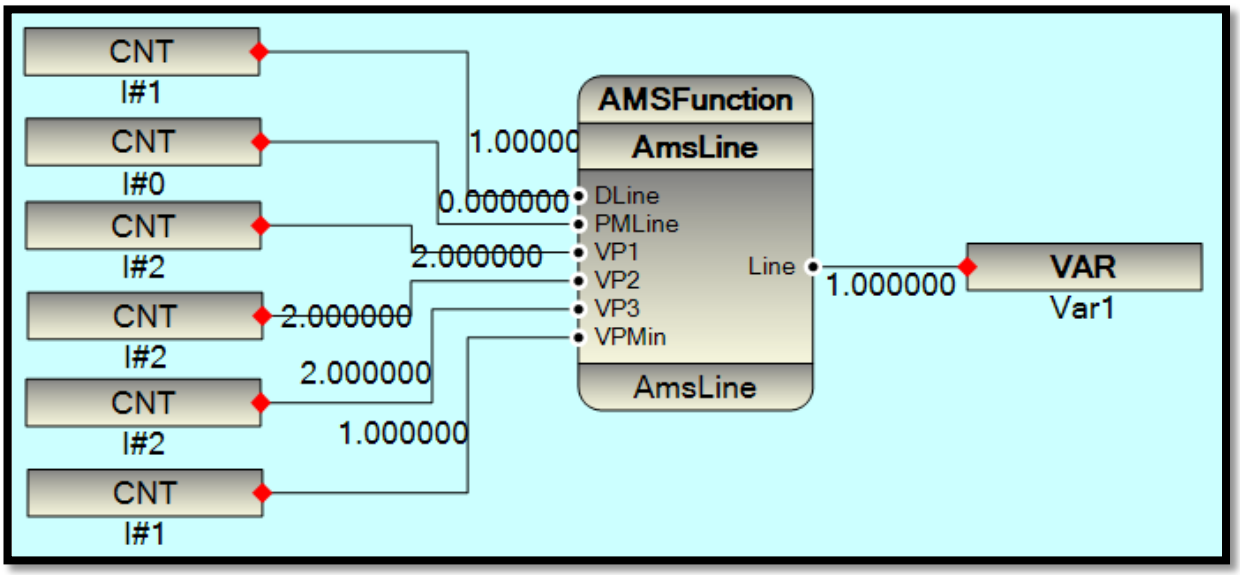


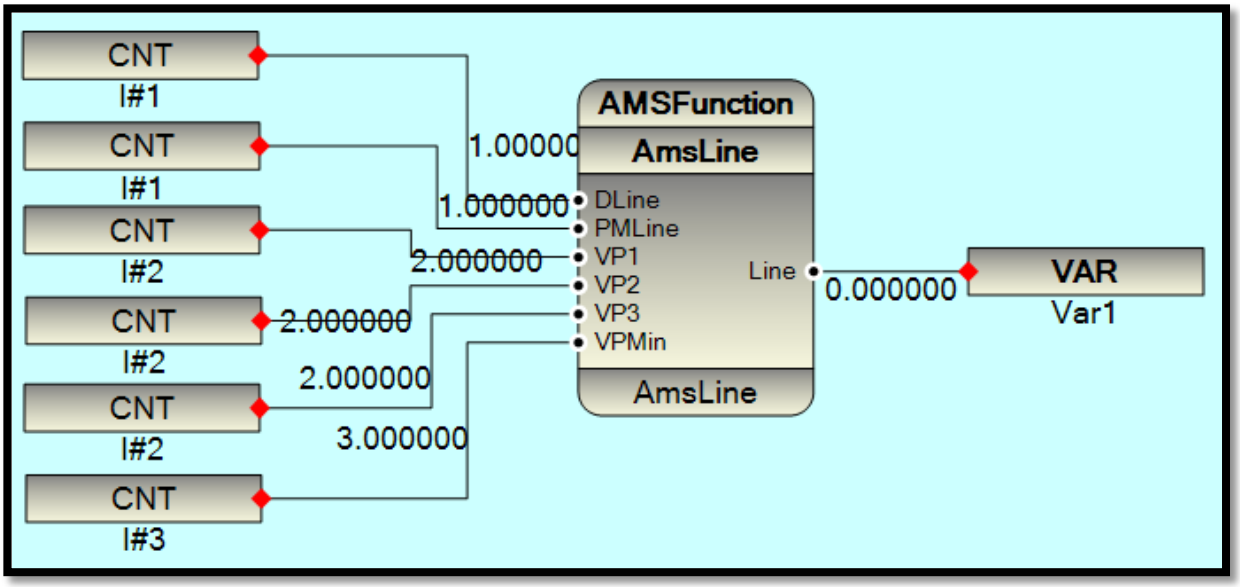
21.8.8 AmsLine Function Block

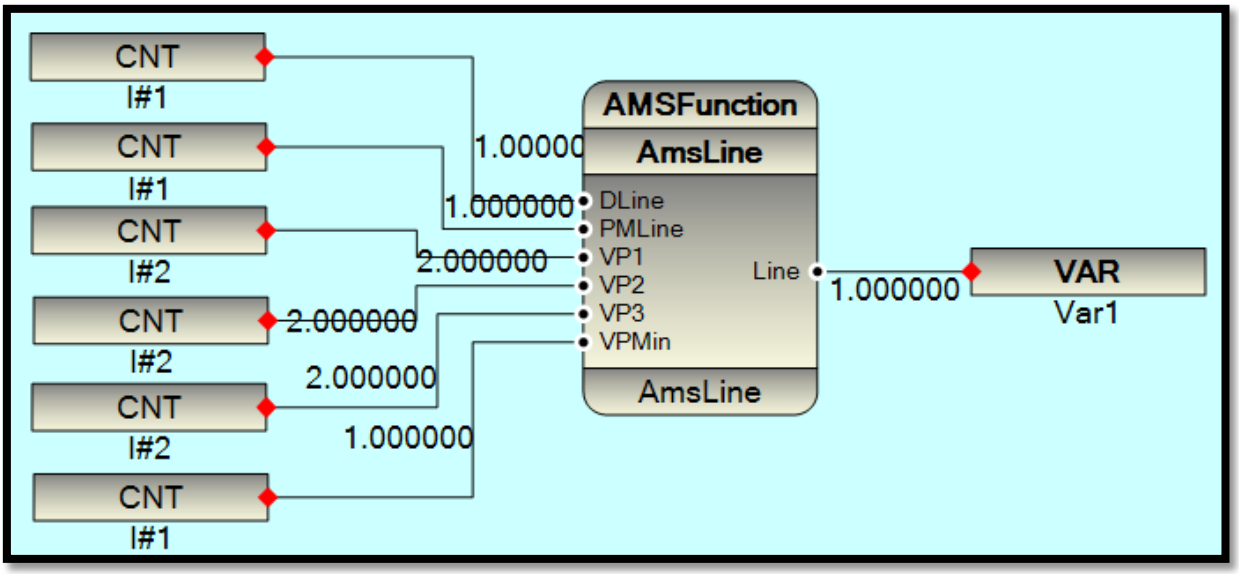


AmsLine		
I/O	Data Type	Description
DLine	Double	
PMLine	Double	
VP1	Double	
VP2	Double	
VP3	Double	
VPMin	Double	
Line	Double	

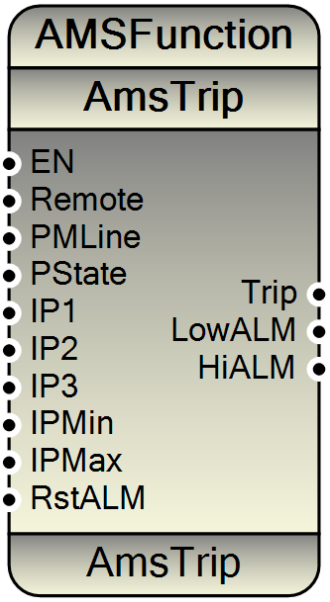



CNT
I#1
CNT
I#0
CNT
I#2
CNT
I#2
CNT
I#2
CNT
I#1
AMSFunction
AmsLine
DLine
PMLine
VP1
VP2
VP3
VPMin
Line
AmsLine
1.00000
0.000000
2.000000
2.000000
2.000000
1.000000
1.000000
VAR
Var1

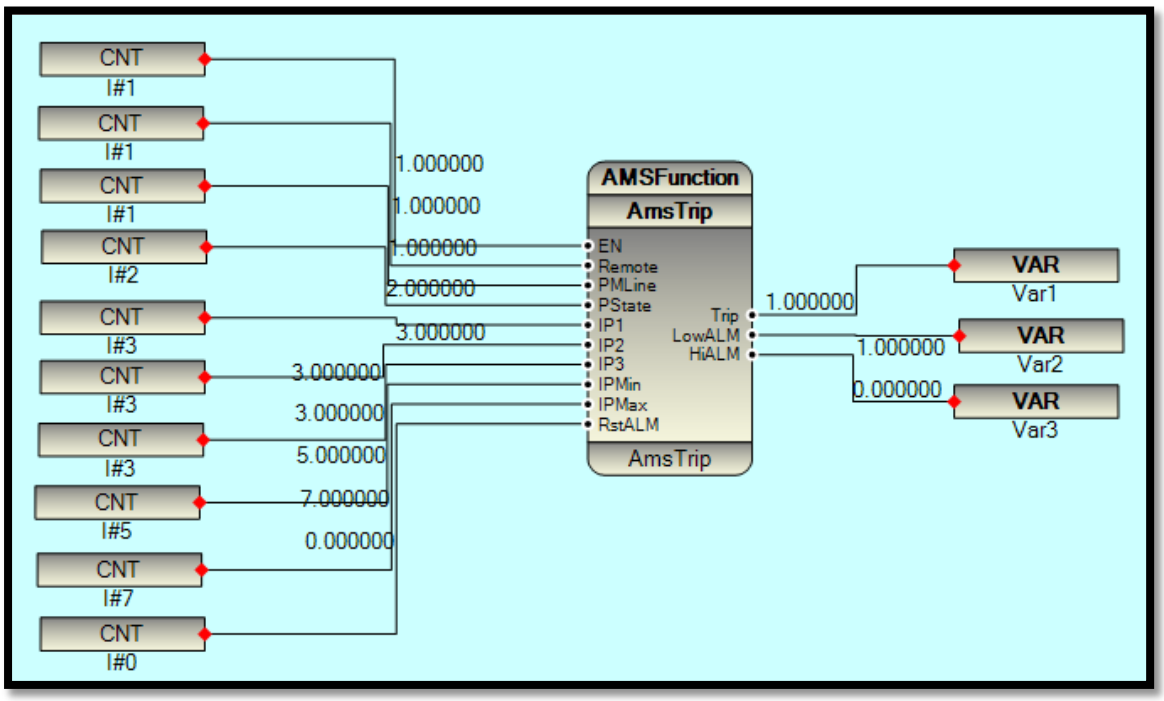

CNT
I#1
CNT
I#1
CNT
I#2
CNT
I#2
CNT
I#2
CNT
I#3
AMSFunction
AmsLine
DLine
PMLine
VP1
VP2
VP3
VPMin
Line
AmsLine
1.00000
1.000000
2.000000
2.000000
2.000000
3.000000
0.000000
VAR
Var1

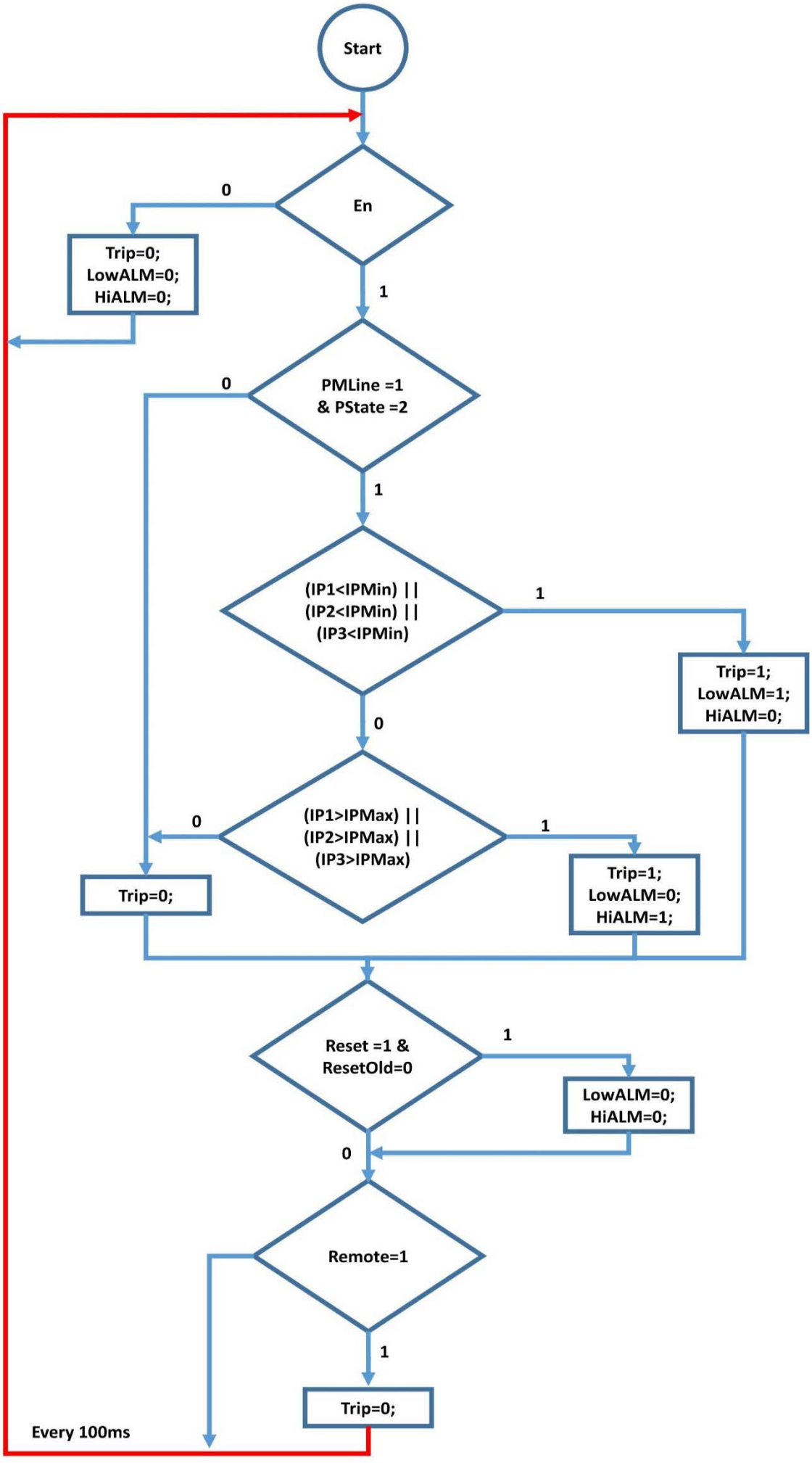

CNT
I#1
CNT
I#1
CNT
I#2
CNT
I#2
CNT
I#2
CNT
I#1
AMSFunction
AmsLine
DLine
PMLine
VP1
VP2
VP3
VPMin
Line
AmsLine
1.00000
1.000000
2.000000
2.000000
2.000000
1.000000
1.000000
VAR
Var1

21.8.9 AmsTrip Function Block

<table>
<tr><th colspan="4">AmsTrip</th></tr>
<tr><td rowspan="13"> 
</td><th>I/O</th><th>Data Type</th><th>Description</th></tr>
<tr><td>En</td><td>Double</td><td></td></tr>
<tr><td>Remote</td><td>Double</td><td></td></tr>
<tr><td>PMLine</td><td>Double</td><td></td></tr>
<tr><td>PState</td><td>Double</td><td></td></tr>
<tr><td>IP1,…,3</td><td>Double</td><td></td></tr>
<tr><td>IPMin</td><td>Double</td><td></td></tr>
<tr><td>IPMax</td><td>Double</td><td></td></tr>
<tr><td>RstALM</td><td>Double</td><td></td></tr>
<tr><td>Trip</td><td>Double</td><td></td></tr>
<tr><td>LowALM</td><td>Double</td><td></td></tr>
<tr><td>HiALM</td><td>Double</td><td></td></tr>
</table>

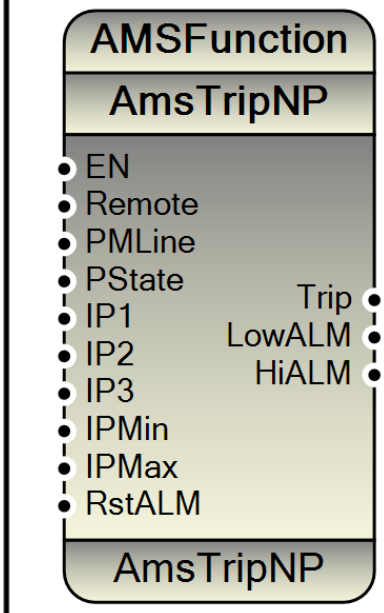
For Example:



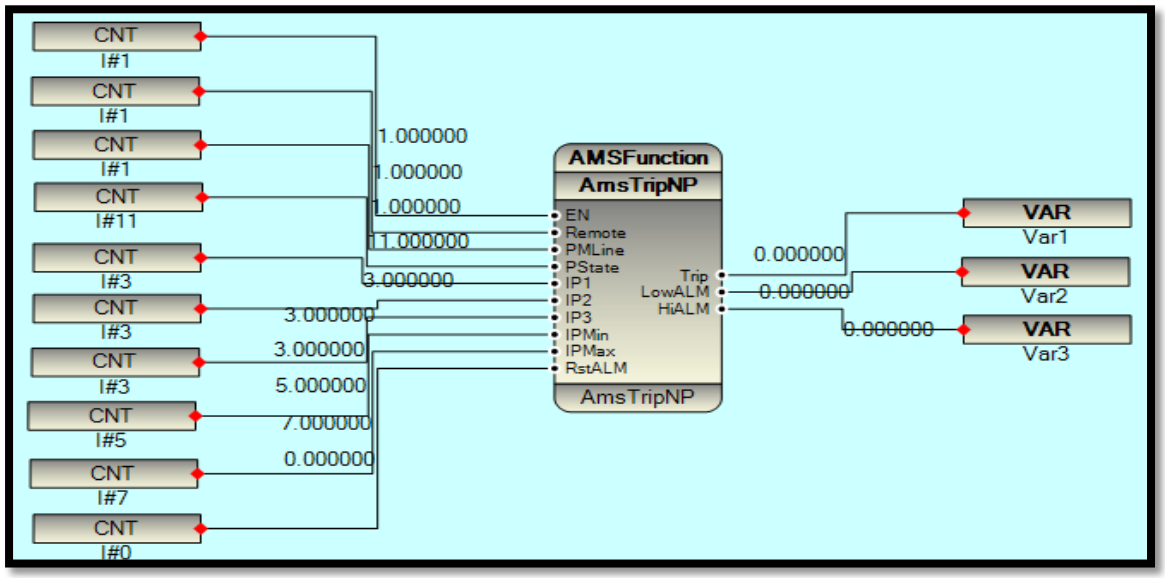

Start
En
0
Trip=0;
LowALM=0;
HiALM=0;
1
PMLine =1
& PState =2
0
1
(IP1<IPMin) ||
(IP2<IPMin) ||
(IP3<IPMin)
1
Trip=1;
LowALM=1;
HiALM=0;
0
(IP1>IPMax) ||
(IP2>IPMax) ||
(IP3>IPMax)
0
1
Trip=0;
Trip=1;
LowALM=0;
HiALM=1;
Reset =1 &
ResetOld=0
1
LowALM=0;
HiALM=0;
0
Remote=1
1
Trip=0;
Every 100ms

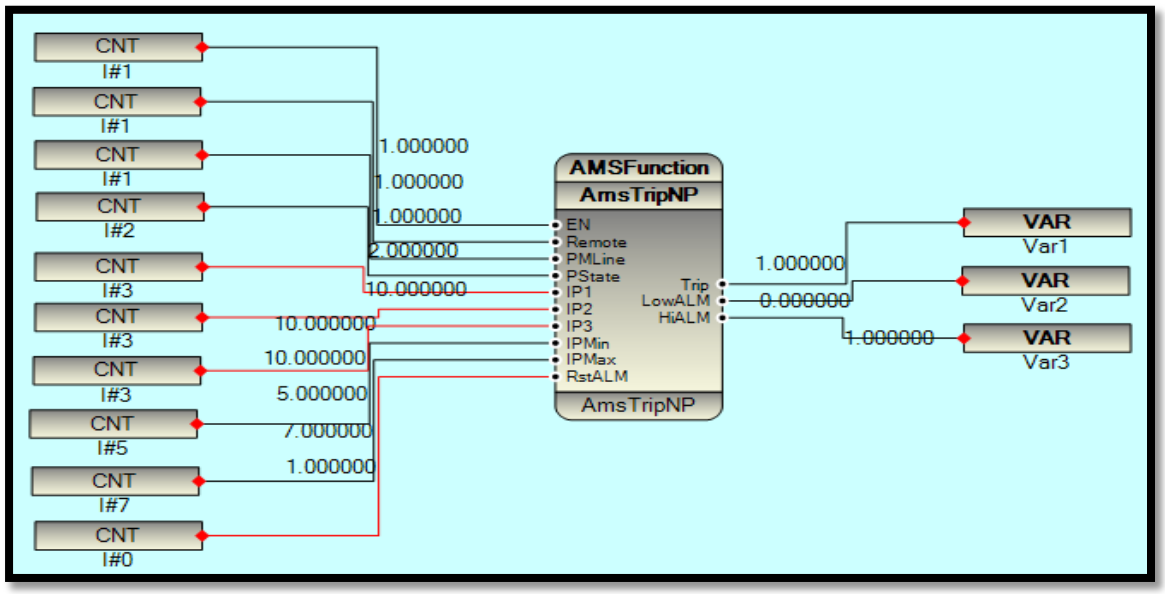
21.8.10 AmsTripNP Function Block

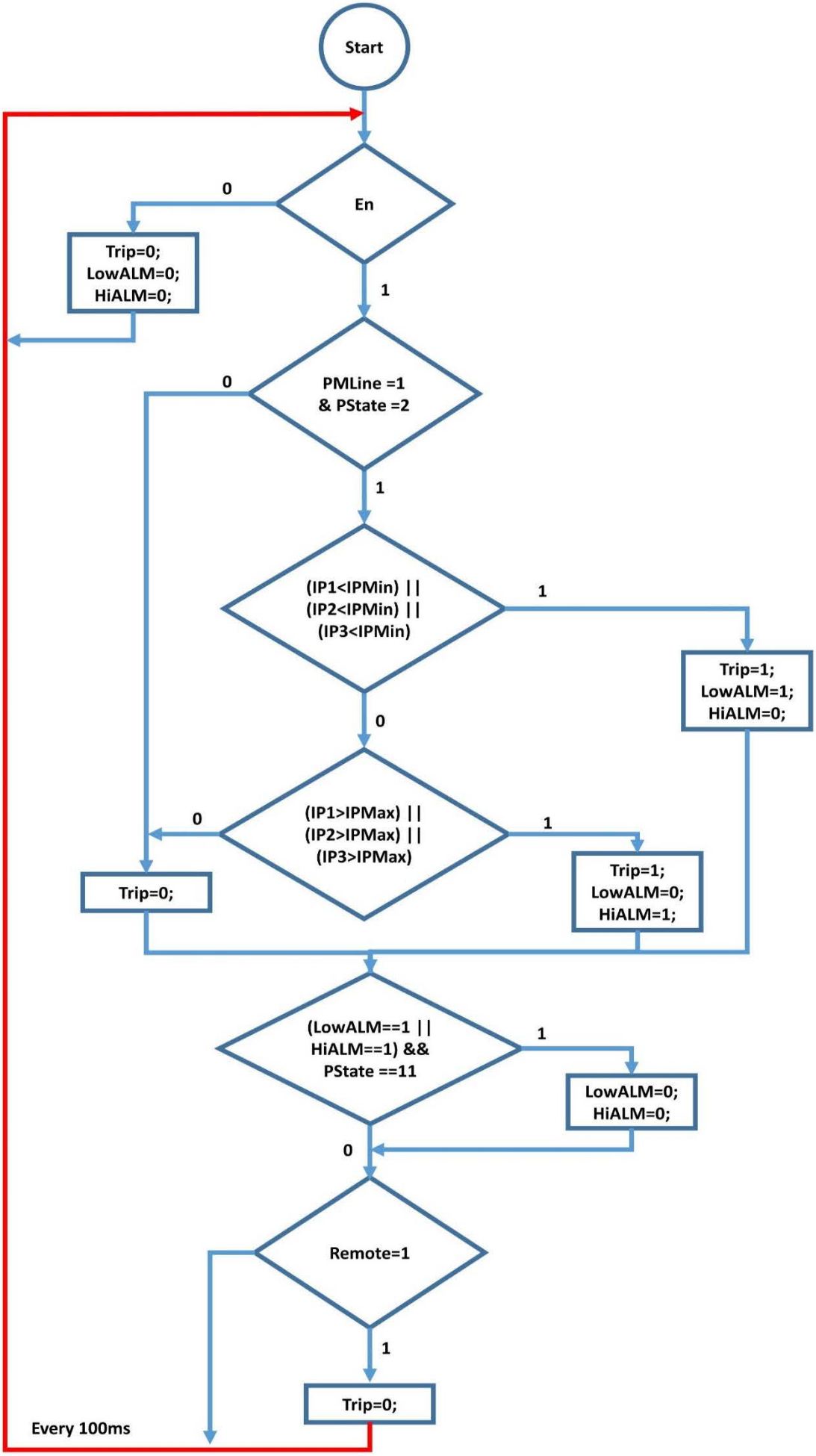
AmsTripNP



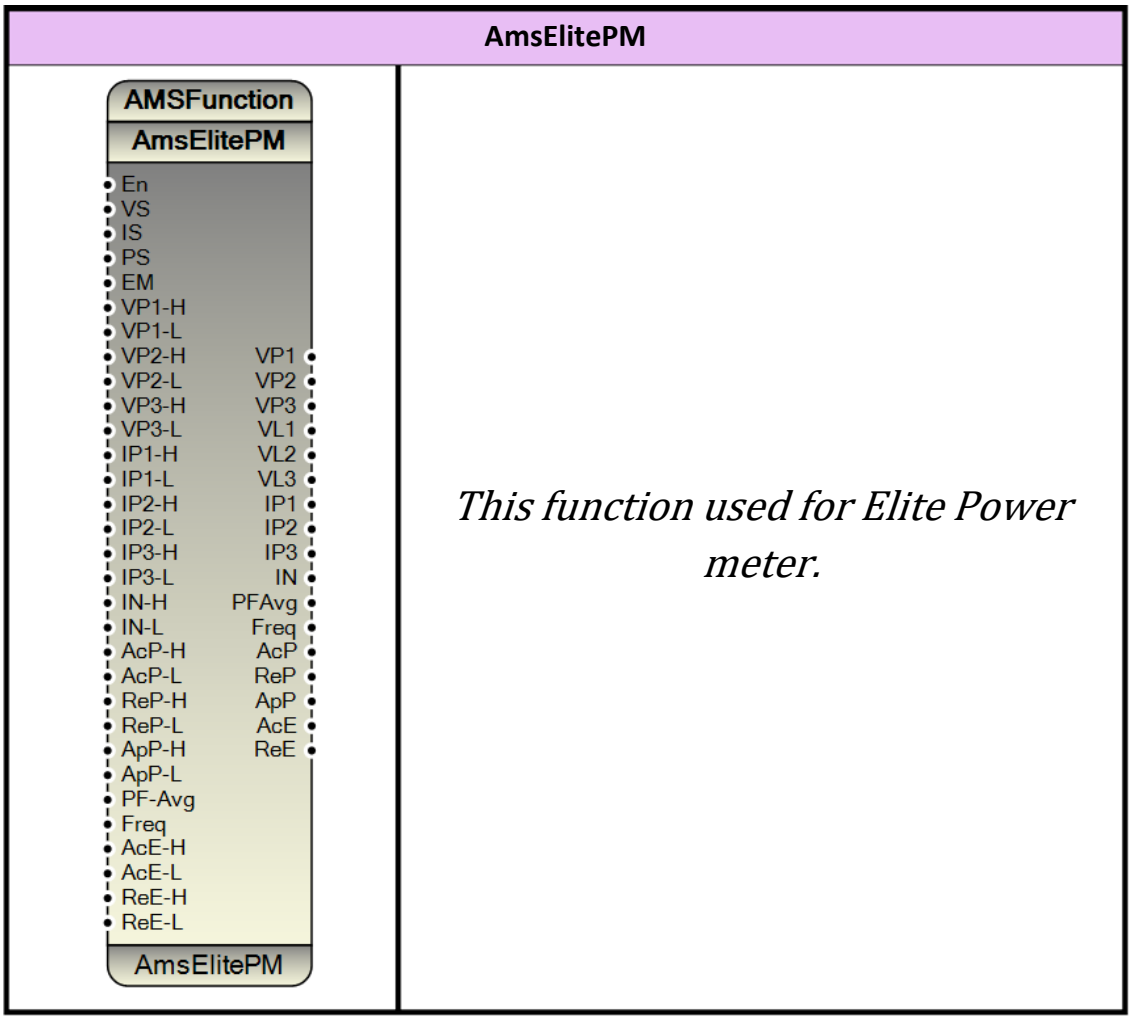
I/O	Data Type	Description
En	Double	
Remote	Double	
PMLine	Double	
PState	Double	
IP1,…,3	Double	
IPMin	Double	
IPMax	Double	
RstALM	Double	NC
Trip	Double	
LowALM	Double	
HiALM	Double	





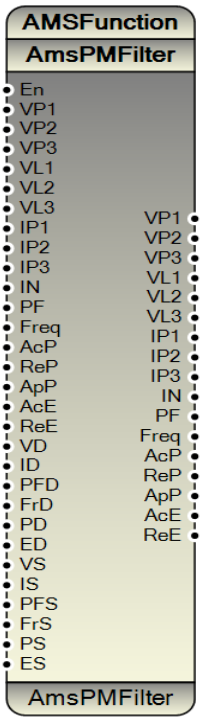

Start
En
0
Trip=0;
LowALM=0;
HiALM=0;
1
PMLine =1
& PState =2
0
1
(IP1<IPMin) ||
(IP2<IPMin) ||
(IP3<IPMin)
1
Trip=1;
LowALM=1;
HiALM=0;
0
(IP1>IPMax) ||
(IP2>IPMax) ||
(IP3>IPMax)
0
1
Trip=0;
Trip=1;
LowALM=0;
HiALM=1;
(LowALM==1 ||
HiALM==1) &&
PState ==11
1
LowALM=0;
HiALM=0;
0
Remote=1
1
Trip=0;
Every 100ms

21.8.11 AmsElitePM Function Block

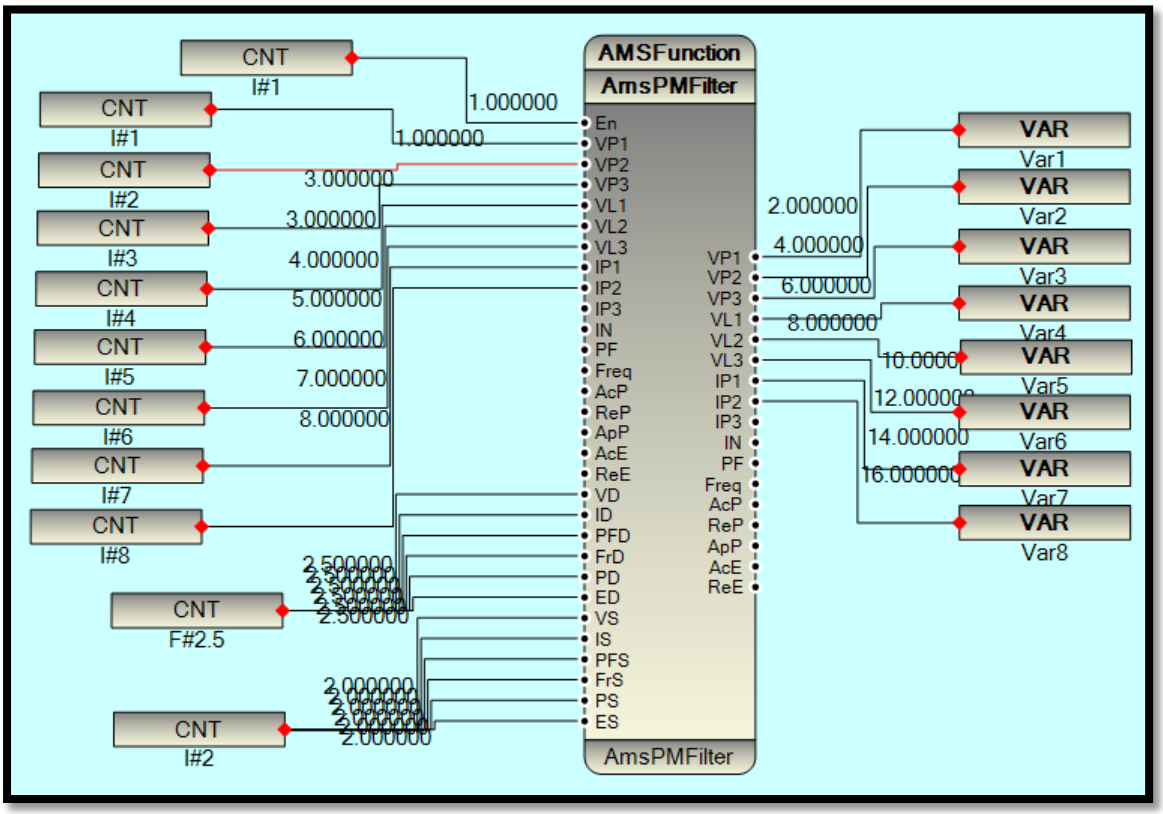


21.8.12 AmsPMFilter Function Block

This function block includes of 17 filters. When Input Signal Changes is more than Delta (VP1, ReE) then Current value of Signal multiplied in scale and will pass to Q Output.In above example when Input signal changes is more than 0.5 then Input Signal multiplied in scale and will map to Output Signal, otherwise old value will pass to output.

AmsPMFilter			
	**I/O**	**Data Type**	**Description**
	En	**Double**	*Enable/Disable*
	VP1,…,ReE	**Double**	*Input Signal*
	VD,…,ED	**Double**	*Tolerance of error*
	VS,…,ES	**Double**	*Scale factor*
	VP1,…,ReE	**Double**	*Output=Scale * In*

For Example:

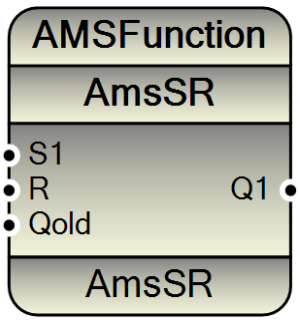


- **21.8.13 AmsPumpMang Function Block**
- **21.8.14 AmsCheckVLV Function Block**
- **21.8.15 AmsCheckSW Function Block**

21.8.16 AmsSR Function Block

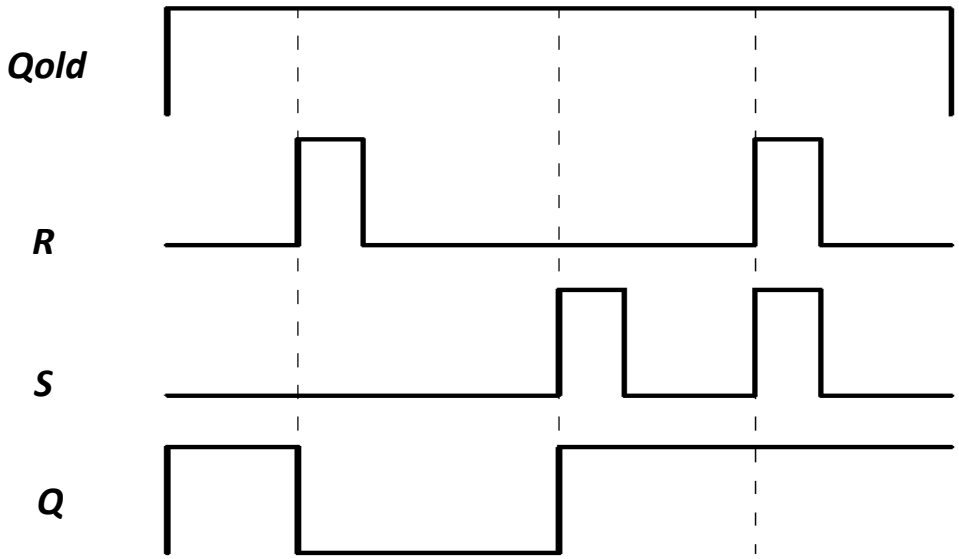
Set Reset Flip Flop.
In time t=0, Q will set to Qold
When R Input changed from 0 to 1: Q will set to 0
When S Input Changed from 0 to 1: Q will set to 1
If R and S changed from 0 to 1 at same time, Q will set to 1

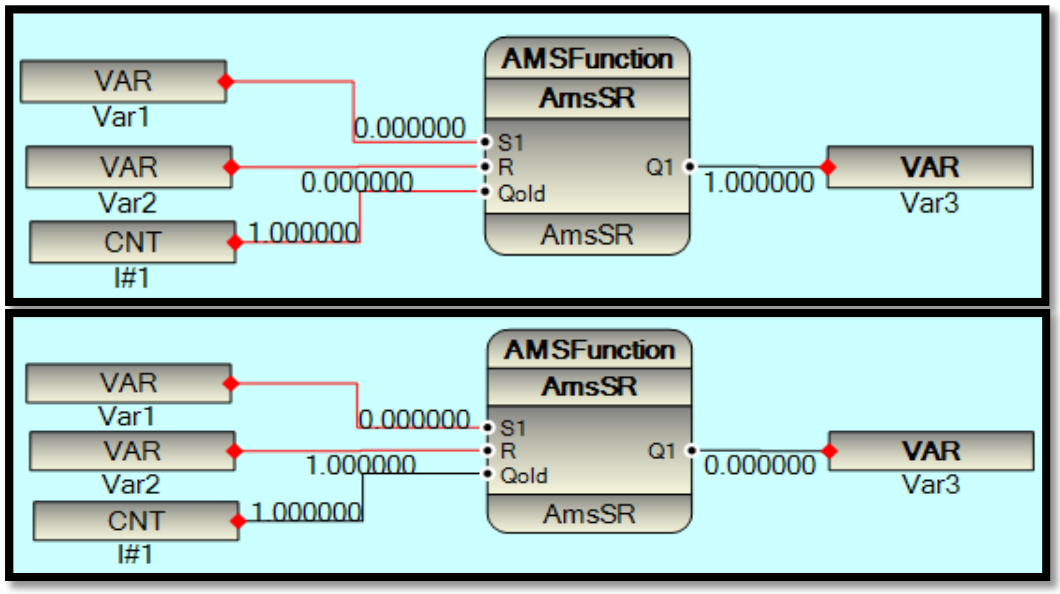
AmsSR



I/O	Data Type	Description
S1	Double	*Set*
R	Double	*Reset*
Qold	Double	*Q(t=0)=Qold*
Q1	Double	*Output*

Logic Diagrams

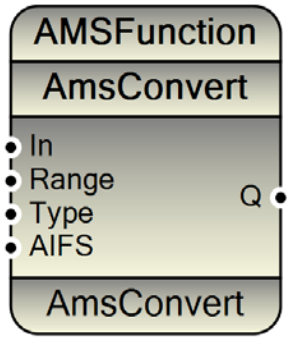




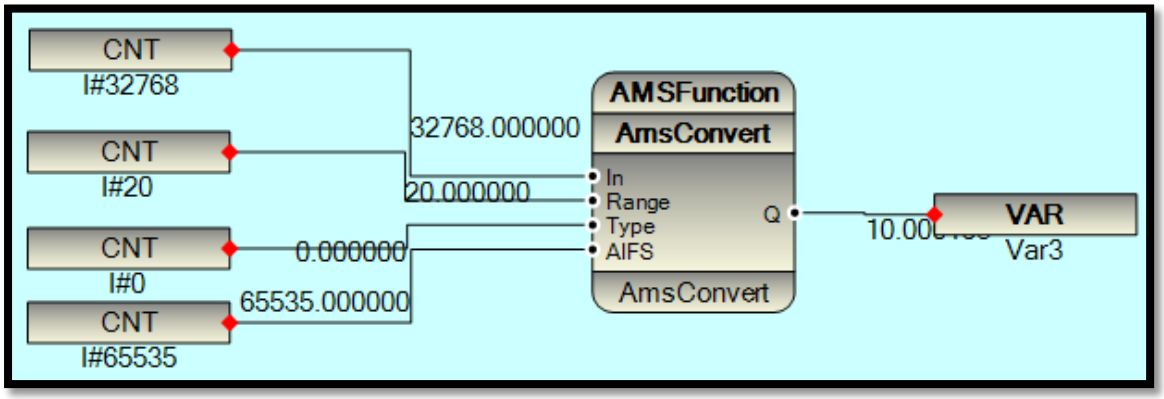
21.8.17 AmsConvert Function Block

This Function Block is Scaling Input Signal.

$$Q = Range \times \frac{In - Type}{AIFS - Type}$$

AmsConvert			
	**I/O**	**Data Type**	**Description**
	In	**Double**	*input*
	Range	**Double**	*Maximum value*
	Type	**Double**	*Minimum value*
	AIFS	**Double**	*Full Scale*

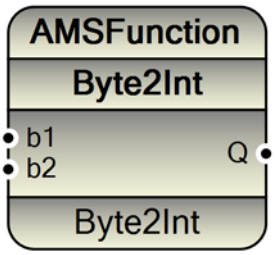
For Example: $20 \times \frac{32768-0}{65535-0} = 10$



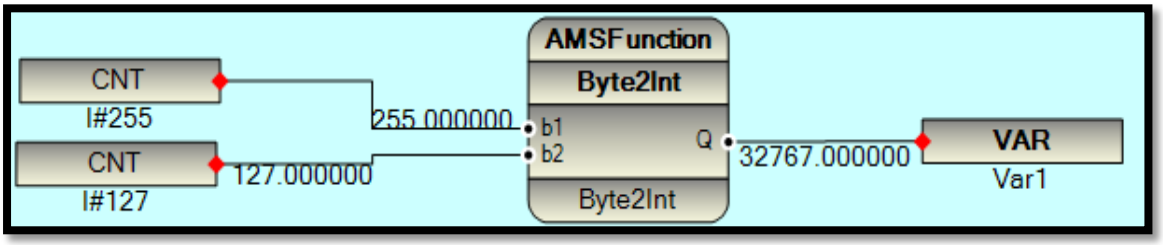
21.8.18 AmsValveV3 Function Block

21.8.19 Byte2Int Function Block

This Function Block is converter byte numbers to Integer numbers.

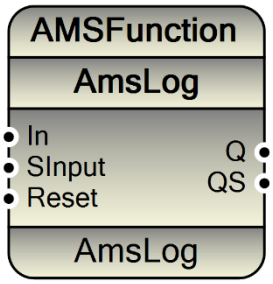
Byte2Int			
	**I/O**	**Data Type**	**Description**
	in1	**Double**	*(least valuable)Byte0*
	in2	**Double**	*Byte1*
	Q	**Double**	*Long numbers(32bit)*

For Example: $(0x7F.0xFF) = 0x7FFF$



21.8.20 AmsLog Function Block

This Function Block, putting SInput in QS at t=0, then summing in and SInput, in the final put result in Q and QS.

AmsLog			
	**I/O**	**Data Type**	**Description**
	In	**Double**	*Input value*
	SInput	**Double**	*Input value(t=0)*
	Reset	**Double**	*Reset*
	Q	**Double**	*Output value*
	QS	**Double**	*Output value(t=0)*