Function Block Reference

D800018X012

EMERSON
Process Management

Printed in USA

DeltaV, the DeltaV design, and PlantWeb are marks of one of the Emerson Process Management group of companies. All other marks are property of their respective owners. The contents of this publication are presented for informational purposes only, and while every effort has been made to ensure their accuracy, they are not to be construed as warrantees or guarantees, expressed or implied, regarding the products or services described herein or their use or applicability. All sales are governed by our terms and conditions, which are available on request. We reserve the right to modify or improve the design or specification of such products at any time without notice.

Contents

Function Blocks - General Information

This book provide detailed information on all of the DeltaV function blocks. This includes schematic diagrams, block executions, application information, parameters, modes, status handling, and alarms.

Function Blocks

The DeltaV system utilizes modular configuration for developing a control strategy. The control modules are treated as unique, named control entities in the DeltaV system. The function block is the basic component of a control module, that is, it is the building block of the control module. Each function block contains standard process control algorithms (such as PID, Analog Out, and Analog In). Advanced process control algorithms are also found in function blocks, such as the DeltaV system's fuzzy logic PID. When wired together in an appropriate and logical sequence, multiple function blocks form a control module. Refer to the Function Block Diagram topic for more information.

General Function Block Information

There are six categories of function blocks in the DeltaV system:

- **Input/Output (I/O) Blocks** – Scale, convert, and filter input and output signals for use in other function blocks or field devices
- **Math Blocks** – Perform mathematical functions for conversion, integration, and totalizing
- **Timer/Counter Blocks** – Perform timing and counting functions for control and sequencing
- **Logical Blocks** – Perform logic functions for sequencing, scheduling, and interlocking
- **Analog Control Blocks** – Perform simple and complex algorithms for comprehensive analog control
- **Energy Metering Blocks** – Perform mathematical flow calculations for natural gases, steam, and other fluids
- **Advanced Control Blocks** – Perform complex algorithms for advanced process control

In addition, there is a collection of special items that contains input and output parameters, internal read and write parameters, and custom and physical block tools.

Guide to Function Blocks

The following table includes the DeltaV palette icon for each function block and a brief description of the capabilities of the block:

DeltaV Standard Function Blocks

Icon	Function Block	Palette Category	Description
	Action (ACT)	Logical Blocks	Evaluates a single-line expression based on an input value. Mathematical functions, logical operators, and constants can be used in the expression.
Σ	Add (ADD)	Math Blocks	Sums the values of two to sixteen inputs and generates an output value. The block supports signal status propagation.
	Alarm Detection (ALARM)	Analog Control Blocks	Provides the ability to easily specify alarms on parameters that are obtained from I/O or from the results of other function block calculations.
	Analog Input (AI)	I/O Blocks	Accesses a single analog measurement value and status from an I/O channel. The input value can be a transmitter's 4 to 20 mA signal or the digitally communicated primary or non-primary variable from a HART transmitter. The block supports mode control, block alarming, signal scaling, signal filtering, signal status calculation, and simulation.
	Analog Output (AO)	I/O Blocks	Assigns an analog output value to a field device through a specified I/O channel. The block supports mode control, signal status calculation, and simulation.
	And (AND)	Logical Blocks	Generates a discrete output value based on the logical AND of two to sixteen discrete inputs. The block supports signal status propagation.
√Σ	Arithmetic	Math Blocks	Provides the ability to configure a range extension function for a primary input and applies the nine different arithmetic types as compensation to or augmentation of the range extended input.

Icon	Function Block	Palette Category	Description
	Bias/Gain (BG)	Analog Control Blocks	Provides adjustable gain capability by computing an output value from a bias setpoint, an input, and a gain value The block supports output tracking.
	Bi-directional Edge (BDE) Trigger	Logical Blocks	Generates a True (1) discrete pulse output when the discrete input makes a positive (False-to-True) or negative (True-to-False) transition since the last execution of the block. The block supports signal status propagation.
	Boolean Fan Input (BFI)	Logical Blocks	Generates a discrete output based on the weighted binary sum, binary coded decimal (BCD) representation, transition state, or logical OR of one to sixteen discrete inputs. The block supports signal status propagation.
	Boolean Fan Output (BFO)	Logical Blocks	Decodes a binary weighted floating point input to individual bits and generates a discrete output value for each bit (as many as sixteen outputs). The block supports signal status propagation.
	Calculation/Logic (CALC)	Analog Control Blocks	Allows you to specify an expression that determines the block's output. Mathematical functions, logical operators, constants, parameter references, and I/O reference values can be used in the expression.
	Comparator (CMP)	Math Blocks	Compares two values and sets a Boolean output based on that comparison.
	Condition (CND)	Logical Blocks	Evaluates a single-line expression and generates a discrete output value when the expression is evaluated True (1) for longer than a specified time period. Mathematical functions, logical operators, and constants can be used in the expression.
	Control Selector (CTLSL)	Analog Control Blocks	Selects one of three control signals to perform override control to a PID function block. The block supports mode control.

Icon	Function Block	Palette Category	Description
001	Counter (CTR)	Timer/Counter Blocks	Generates a discrete output of True (1) when the count reaches a specified trip value. The block functions as an incremental (up) counter or a decremental (down) counter. The block supports signal status propagation.
	Date Time Event (DTE)	Timer/Counter Blocks	Provides Time Of Day Timer functions. Use the Timer functions to obtain date and time information in various formats, as well as manipulate date and time information. The function block allows scheduling future events at a specified date and time.
	Deadtime (DT)	Analog Control Blocks	Introduces a pure time delay in the value and status used in a signal path between two function blocks. The block supports signal status propagation and mode control.
	Device Control (DC)	Logical Blocks	Provides setpoint control for multistate discrete devices. As many as four inputs and four outputs can be controlled. The basic functionality is augmented by an assortment of interlocks and device control options to customize the block's operation to your application. The block supports mode control, simulation, field device confirmation, and alarm limit detection.
	Diagnostic (DIAG)	Advanced Control Blocks	Provides a method to monitor device alerts from non-fieldbus assets. Wire parameters or block outputs that indicate device health for non-fieldbus blocks to the Diagnostic block. These alerts from these diagnostic blocks are monitored by the Inspect application.
	Discrete Input (DI)	I/O Blocks	Accesses a single discrete measurement value and status from a two-state field device and makes the processed physical input available to other function blocks. The block supports mode control, signal status propagation, and simulation.

Icon	Function Block	Palette Category	Description
	Discrete Output (DO)	I/O Blocks	Takes a binary setpoint and writes it to a specified I/O channel to produce an output signal. The block supports mode control, output tracking, simulation, and field device confirmation.
	Divide (DIV)	Math Blocks	Divides one input value by another input value and generates an output value. The block supports signal status propagation.
	Filter (FLTR)	Analog Control Blocks	Applies an equation to filter changes in the input signal and generates a smooth output signal. The block supports signal status propagation.
	Flow Metering (AGA_SI and AGA_US)	Energy Metering Blocks	Implements the American Gas Association flow calculation standards for natural gases, namely AGA-3 (American Gas Association, Report No.3), AGA-7, and AGA-8.
	Fuzzy Logic Control (FLC)	Advanced Control Blocks	Provides all the logic to perform standard PID control with the added benefit of superior response for both set point changes and external load disturbances.
	Input Selector (INSEL)	Analog Control Blocks	A mathematical and logical input calculation block that chooses an output based on up to 4 inputs.
	Input Selector Extended (ISELX)	Analog Control Blocks	A mathematical and logical input calculation block that chooses an output based on up to 8 inputs.
	Inspect	Advanced Control Blocks	A block that receives statistics data so that you can view, plot, and add to history, the performance values in the system.

Icon	Function Block	Palette Category	Description
	Integrator (INT)	Math Blocks	Integrates a variable over time. The block compares the integrated or accumulated value to pre-trip and trip limits and generates discrete output signals when the limits are reached. The integrated value can increment from zero or decrement from the trip value. The block has two inputs and can calculate and integrate net flow, as well as handling negative flow. The block supports mode control.
	Isentropic Expansion (ISE)	Energy Metering Blocks	Calculates the final enthalpy for isentropic expansion of steam to a given pressure for a given entropy.
	Lab Entry (LE)	Advanced Control Blocks	Provides for operator input of offline lab analysis results.
	Lead/Lag (LL)	Analog Control Blocks	Provides dynamic compensation for an input value. The block can apply a lead time function, a lag time function, or a combination of the two. A specified gain is applied to the compensated value, and the value is high/low-limited based on the block mode. The block supports mode control and signal status propagation.
	Limit (LIM)	Analog Control Blocks	Limits an input value between two reference values. The block supports signal status propagation.
	Manual Loader (MANLD)	Analog Control Blocks	Allows the block output to be set by an operator. The block supports output tracking and alarm detection.
	Model Predictive Control (MPC)	Advanced Control Blocks	Allows interactive processes to be controlled within measurable operating constraints while automatically accounting for process interaction and measurable disturbances.

Icon	Function Block	Palette Category	Description
	Model Predictive Control Professional (MPCPro)	Advanced Control Blocks	Allows large interactive processes (as large as 20 x20) to be controlled within measurable operating constraints while accounting for process interaction and measurable disturbances. An embedded optimizer is included with the MPCPro block that can be used to effectively provide maximum profit or lowest cost production with the process constraints and limits on process inputs.
	MPC Process Simulator	Advanced Control Blocks	Simulates the actual process for use with the MPC function block for operator training.
	Multiple Discrete Input (FFMDI)	I/O Blocks	Combines the eight channels of a Discrete Input card and makes them available as an 8-bit input to other function blocks.
	Multiple Discrete Output (FFMDO)	I/O Blocks	Takes an 8-bit setpoint and writes it to the I/O channels of a Discrete Output card on an H1 Carrier device.
	Multiplexed Analog Input (MAI)	I/O Blocks	Connects higher density transmitters to a Fieldbus segment.
	Multiplexer (MLTX)	Logical Blocks	Selects one input value from as many as sixteen input values and places it at the output. The block supports signal status propagation.
	Multiply (MLTY)	Math Blocks	Multiplies two to sixteen input values and generates an output value. The block supports signal status propagation.
	Negative Edge Trigger (NDE)	Logical Blocks	Generates a True (1) discrete pulse output when the discrete input makes a negative (True-to-False) transition since the last execution of the block. The block supports signal status propagation.
	Neural Network (NN)	Advanced Control Blocks	Uses a neural network to predict a process output based on measured process inputs.

Icon	Function Block	Palette Category	Description
	Not (NOT)	Logical Blocks	Logically inverts a discrete input signal and generates a discrete output value. The block supports signal status propagation.
	Off-delay Timer (OFFD)	Timer/Counter Blocks	Delays the transfer of a False (0) discrete input value to the output by a specified time period. The block supports signal status propagation.
	On-delay Timer (OND)	Timer/Counter Blocks	Delays the transfer of a True (1) discrete input value to the output by a specified time period. The block supports signal status propagation.
	Or (OR)	Logical Blocks	Generates a discrete output value based on the logical OR of two to sixteen discrete inputs. The block supports signal status propagation.
K∫δ	PID (PID)	Analog Control Blocks	Combines all the necessary logic to perform analog input channel processing, proportional-integral-derivative (PID) control, and analog output channel processing within one function block. The block supports mode control, signal scaling and limiting, feedforward control, override tracking, alarm limit detection, and signal status propagation.
	Positive Edge Trigger (PDE)	Logical Blocks	Generates a True (1) discrete pulse output when the discrete input makes a positive (False-to-True) transition since the last execution of the block. The block supports signal status propagation.
	Pulse Input (PIN)	I/O Blocks	Provides analog input values from Pulse Input channels on the Multifunction I/O card.
	Ramp (RAMP)	Analog Control Blocks	Creates a ramping output signal to increase or decrease a variable toward a specified target value at a defined rate. The block supports signal status propagation.

Icon	Function Block	Palette Category	Description
V▷	Rate Limit (RTLM)	Analog Control Blocks	Limits the rate of change of the output value to specified limits. The block supports signal status propagation.
1:1	Ratio (RTO)	Analog Control Blocks	Applies an adjustable ratio setpoint to achieve a desired input/output relationship. The block supports signal filtering, mode control, output tracking, and alarm detection.
S R	Reset/Set Flip-flop (RS)	Logical Blocks	Generates a discrete output value based on NOR logic of reset and set inputs.
	Retentive Timer (RET)	Timer/Counter Blocks	Generates a True (1) discrete output after the input has been True for a specified time period. The elapsed time the input has been True and the output value are reset when the reset input is set True.
20 100% 4 0%	Scaler (SCLR)	Analog Control Blocks	Provides scaling and dimensional consistency between two values of different engineering units. The block converts the input value to the specified scale and generates an output value. The block supports signal status propagation.
S R	Set/Reset Flip-flop (SR)	Logical Blocks	Generates a discrete output value based on NAND logic of set and reset inputs.
	Signal Characterizer (SGCR)	Analog Control Blocks	Characterizes or approximates any function that defines an input/output relationship. The block interpolates an output value for a given input value using the curve defined by the configured coordinates. Two separate analog input signals can be processed simultaneously to give two corresponding separate output values using the same defined curve. The block supports signal status propagation.

Icon	Function Block	Palette Category	Description
	Signal Generator (SGGN)	Analog Control Blocks	Produces an output signal used to simulate a process signal. The block uses a specified combination of a sine wave, a square wave, a bias value, and a random value to generate the output signal.
	Signal Selector (SGSL)	Analog Control Blocks	Selects the maximum, minimum, or average of as many as sixteen input values and places it at the output. The block supports signal status propagation.
	Splitter (SPLTR)	Analog Control Blocks	Takes a single input and calculates two outputs based on specified coordinate values. The block supports mode control and signal status propagation.
	Subtract (SUB)	Math Blocks	Subtracts one input value from another input value and generates an output value. The block supports signal status propagation.
	Steam Density Ratio (SDR)	Energy Metering Blocks	Calculates the square root of the ratio of steam density to the density of steam corresponding to a flow meter calibration pressure and temperature.
	Saturated Steam Properties at Temperature (SST)	Energy Metering Blocks	Calculates steam enthalpy, entropy, specific volume, and pressure for saturation conditions specified by a given temperature.
	Steam Properties (STM)	Energy Metering Blocks	Calculates steam enthalpy, entropy, and specific volume for a given gauge pressure.
	Saturated Temperature (TSS)	Energy Metering Blocks	Calculates the steam temperature at saturation given the steam pressure.
	Timed Pulse (TP)	Timer/Counter Blocks	Generates a True (1) discrete output for a specified time duration when the input makes a positive (False-to-True) transition. The output remains True even when the input returns to its initial discrete value and returns to its original False value only when the output is True longer than the specified time duration.

Icon	Function Block	Palette Category	Description
	Transfer (XFR)	Logical Blocks	Selects one of two analog input signals and transfers the selected input to the output after a specified time. The transfer from one input to another is smoothed with a linear ramp. The block supports signal status propagation.
	Water Entropy (WTS)	Energy Metering Blocks	Calculates the entropy of water for a specified temperature.
	Water Enthalpy (WTH)	Energy Metering Blocks	Calculates the enthalpy of water for a specific temperature.

What is a Function Block?

A function block is a type of building block for creating the algorithms that perform the actual control or monitoring for your process. Each function block contains an algorithm and parameters that customize the algorithm. Function block algorithms range from simple input conversions to complex control strategies. The function block uses parameter data supplied by the user, by the function block itself, or by other function blocks to perform its calculations and logic functions and to supply an output value to other function blocks or to field devices. Some function blocks also detect alarm conditions.

You can connect function blocks together so that data can be transmitted between blocks. This data can be used in the control algorithm, mathematical or logical calculation, or status determination of the block. Refer to the Function Block Diagram topic for more information. This capability helps you implement a variety of process control strategies, including advanced control, safe shutdown sequences, and process reactions to quality control information.

Extensible Blocks

Many of the standard function blocks are extensible. This capability means you can increase the number of some of the parameters. The function blocks that are extensible are: Add, And, BFI, BFO, Calc/Logic, Multiply, Multiplexer, Or, and Signal Selector.

For example, you do not have to join together multiple ADD blocks to add more than two numbers together. You can extend the number of inputs on the block and wire as many as 16 values into a single ADD block. Refer to the Extensible Parameters topic for more information.

Block Scan Rate

Blocks can execute at different rates within the same module using the block scan rate. The block scan rate forces the function block's execution to skip a certain number of scans for that module. By default, function blocks in a module have a block scan rate of one (1). If the block scan rate is one, it is not displayed on the block. You can change the block's scan rate to a positive integer value.

The block scan rate indicates the number of times the module's algorithm is executed compared to the block. For example, if the block scan rate is one, there is a 1:1 ratio. Every time the module scans, the block executes. If the block scan rate is 3, there is a 1:3 ratio and the block is executed every third scan.

If a module's scan rate is 1 second and you set the block scan rate of a block in that module to 5, the block executes every 5 seconds. The block execution skips for four module scans and executes on the fifth scan. The block scan rate of the block effectively multiplies the module scan rate.

Use this feature for cascade control. For example, combine the two loops required for cascade control into one module, as long as the scan rate for the outer loop is a multiple of the scan rate for the inner loop.

Function Block Modes

Function block modes are described below. Supported modes vary with each function block. Some blocks support modes that are not supported in other blocks.

Out of Service (OOS): The block algorithm is not active. The output is maintained at the last value or at a specified failure action value.

Initialization Manual (IMan): The upstream block of a cascade pair is put into this mode when its downstream partner is in a non-cascade mode. This prevents the upstream block from closing the cascade. When the downstream block returns to a cascade mode (Cas or RCas), the upstream block leaves IMan and returns to its target mode.

Local Override (LO): The block is put into this mode when tracking is activated; the output is driven to a value other than that generated by normal block execution. When tracking is deactivated, the block returns to its target mode.

Manual (Man): The block output is set directly by the operator.

Automatic (Auto): In this mode, the control algorithm of the block is active. An operator-entered setpoint is used in the control algorithm to determine the block output.

Cascade (Cas): This mode is similar to Auto except that the setpoint is supplied by another function block through the CAS_IN parameter. The block maintains a back calculation output value (BKCAL_OUT) to provide bumpless mode transfer when the mode is changed.

Remote Cascade (RCas): This mode is similar to Cas except that the setpoint is supplied by an external control program through the RCAS_IN parameter. The block maintains a back calculation output value (RCAS_OUT) to provide bumpless transfer when the mode is changed.

Remote Out (ROut): This mode is similar to Man except that the OUT value is supplied by an external control program rather than directly by the operator. OUT is supplied through the ROUT_IN parameter. The block maintains a back calculation output value (ROUT_OUT) to provide bumpless transfer when the mode is changed.

The normal mode is the desired operating mode of the block that is set during configuration. Normal is meant to indicate that this is the mode in which the block should be operating during normal operation.

The target mode is the mode that is set manually or by another function during operation. These target modes must be listed during configuration as permitted modes for the block. IMan and LO are not selectable target modes. Unless you specify otherwise during configuration, the system assumes that you will write mode changes to the target mode field and read them from the actual mode field.

The following table lists the supported modes in specific function blocks. In addition, the Books Online information on individual function blocks lists the supported modes for that block.

Supported Modes for Function Blocks

Function Block	Supported Modes
Analog Input Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Analog Output Function Block	Initialization Manual (IMan) Out of Service (OOS) Local Override (LO)* Manual (Man) Automatic (Auto) Cascade (Cas) Remote Cascade (Rcas)
Arithmetic Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Bias/Gain Function Block	Out of Service (OOS) Initialization Manual (IMan) Local Override (LO) Manual (Man) Automatic (Auto) Cascade (Cas) Remote Cascade (Rcas)
Control Selector Function Block	Out of Service (OOS) Initialization Manual (IMan) Manual (Man) Automatic (Auto)
Deadtime Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Device Control Function Block	Out of Service (OOS)* Local Override (LO)* Automatic (Auto) Cascade (Cas) Remote Cascade (RCas)*
Discrete Input Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Discrete Output Function Block	Initialization Manual (IMan) Out of Service (OOS) Local Override (LO)* Manual (Man) Automatic (Auto) Cascade (Cas) Remote Cascade (RCas)*

Function Block	Supported Modes
Fuzzy Logic Control Function Block	Out of Service (OOS) Initialization Manual (IMan) Local Override (LO) Manual (Man) Automatic (Auto) Cascade (Cas) Remote Cascade (RCas) Remote Output (ROut)
Input Selector Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Input Selector Extended Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Integrator Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Lab Entry Function Block	Out of Service (OOS) Manual (Man)
Lead/Lag Function Block	Out of Service (OOS) Manual (Man) Automatic (Auto)
Manual Loader Function Block	Out of Service (OOS) Initialization Manual (IMan) Local Override (LO) Manual (Man)
Model Predictive Control (MPC)	Out of Service (OOS) Initialization Manual (IMan) Manual (Man) Automatic (Auto) Local Override (LO)
Model Predictive Control Professional (MPCPro)	Out of Service (OOS) Initialization Manual (IMan) Manual (Man) Automatic (Auto) Local Override (LO) Cascade
MPC Process Simulator	Out of Service (OOS) Manual (Man) Automatic (Auto)
Multiple Discrete Input	Out of Service (OOS) Manual (Man) Automatic (Auto)

Function Block	**Supported Modes**
Multiple Discrete Output	Out of Service (OOS) Initialization Manual (IMan) Manual (Man) Automatic (Auto) Cascade (Cas) Local Override (LO) Remote Cascade (RCas)
Multiplexed Analog Input	Out of Service (OOS) Manual (Man) Automatic (Auto)
Neural Network	Out of Service (OOS) Manual (Man) Automatic (Auto)
PID Function Block	Out of Service (OOS) Initialization Manual (IMan) Local Override (LO) Manual (Man) Automatic (Auto) Cascade (Cas) Remote Cascade (RCas) Remote Out (ROut)
Ratio Function Block	Out of Service (OOS) Initialization Manual (IMan) Local Override (LO) Manual (Man) Automatic (Auto) Cascade (Cas) Remote Cascade (RCas)
Signal Characterizer Function Block	Out of Service (OOS) Automatic (Auto)
Splitter Function Block	Out of Service (OOS) Initialization Manual (IMan) Manual (Man) Automatic (Auto) Cascade (Cas)

* These modes are only visible when the function block is extended to a Fieldbus device.

Function Block Mode Basics

Function blocks support modes with the relevant fields:

- Target Mode – The mode that the block is trying to attain. When an operator sets the mode, it is the target mode that is being set.
- Actual Mode – The current operating mode of the block.
- Permitted – Those modes allowed as target modes. Write service prevents setting target mode to a non-permitted mode. Permitted mode is configurable by the user for each instance of each block, as defined by the supported mode.
- Normal – The correct mode for most cases of plant operation.

Each block supports a subset of the following modes through the named set parameter, SHED_OPT:

- RCas – Can be a permitted target mode. Remote cascade connection to setpoint. Setpoint is driven through RCAS_IN (the RCas input). The block algorithm determines block outputs.
- ROut – Can be a permitted target mode. Remote cascade connection to output. The block output is driven through ROUT_IN (the ROut input).
- Cas – Can be a permitted target mode. Cascade connection to setpoint. Setpoint is being driven through CAS_IN (the Cas input). The block algorithm determines block outputs.
- Auto – Can be a permitted target mode. Setpoint is local to the block, (not being driven externally). The block algorithm determines block outputs.
- Man – Can be a permitted target mode. Operator or logic external to the function block determines output.
- IMan – Is never a permitted target mode. Original definition was Initialization Manipulation, which means that the block tracks downstream operation (typically another block that has an open cascade).
- LO – Is never a permitted mode. Local Override or locked output. For a control block, the output tracks a specific input (the action triggered by a discrete track switch input). For an output block, failure action initiates.
- OOS – Is always a permitted mode. The block is out of service.

Modes have priority. The above is in inverse priority order. Higher modes have lower priority. For complete descriptions of all of the function block modes, refer to the Function Block Modes topic.

Mode Shedding Logic

Shed from or climb to a remote mode is determined by the parameter SHED_OPT. A block climbs and sheds through the same path. For example, if SHED_OPT specifies that a block should shed to Auto, then, if the block target mode is set to RCas, the block goes through Auto on the way to RCas.

Note Consider the effect of the fault state when selecting the mode shed option. Keep in mind that the fault state can cause the mode to go to Local Override instead of the expected mode from mode shedding.

Configuring Shed

The parameter SHED_OPT determines shed and climb for a remote mode. It can be configured as follows:

Shed With Return Options

Remote cascade connection failure shifts actual mode but keeps trying to restore remote cascade (in other words, the remote cascade target mode stays in effect).

- Normal – On failure of a remote cascade connection, the block attempts to attain the highest permitted non-remote mode until remote cascade is restored. On change to remote cascade target mode from any other mode, the block attempts to attain the highest permitted non-remote mode until a remote cascade connection is established.
- Retained Target – On failure of a remote cascade connection, the block attempts to attain the mode retained in the target mode. On change to the remote cascade target mode from any other mode, the block attempts to attain the mode retained by target mode until a remote cascade connection is established. The DeltaV system does not support the retained target shed. Selecting the retained target results in shed to Auto.
- Auto – On failure of a remote cascade connection, the block attempts to attain Auto, if permitted, until remote cascade is restored. On change to remote target mode from any other mode, the block attempts to attain Auto, if permitted, until a remote cascade connection is established.
- Man – On failure of a remote cascade connection, the block sheds to Man until a remote cascade connection is restored. On change to remote target mode, from any other mode, the block goes to Man until a cascade connection is established.

Shed With No Return Options

For any shed with no return option, the target mode changes as determined by the option. Therefore, there is no attempt to restore the connection following failure. The behavior on change to the remote cascade target mode is identical to that for Shed With Return Options.

- Normal – On failure of a remote cascade connection, the block sets the target mode to the highest permitted non-remote mode. On change to a remote cascade target mode from any other mode, the block attempts to attain the highest permitted non-remote mode until a remote cascade connection is established.
- Retained Target – On failure of a remote cascade connection, the block sets the target mode to mode retained in target mode. On change to the remote cascade target mode from any other mode, the block attempts to attain the mode retained by target mode until a remote cascade connection is established. The DeltaV system does not support the retained target shed. Selecting the retained target results in shed to Auto. This applies to any block, regardless of whether it is in a field device or a DeltaV controller. The writing device sets or clears the retained target bits in the target mode. Therefore, any workstation or other device (DeltaV controller applications, for example) that does not support retained target operation ignores the retained target bits. In a DeltaV system, regardless of where the block is located, the retained target bits are turned off, and the block sheds to Auto.
- Auto – On failure of a remote cascade connection, the block sets the target mode to Auto, if permitted. On change to the remote target mode from any other mode, the block attempts to attain Auto, if permitted, until a remote cascade connection is established.
- Man – On failure of remote cascade connection, the block sets the target mode to Man, if permitted. On change to the remote target mode from any other mode, the block goes to Man until a cascade connection is established.

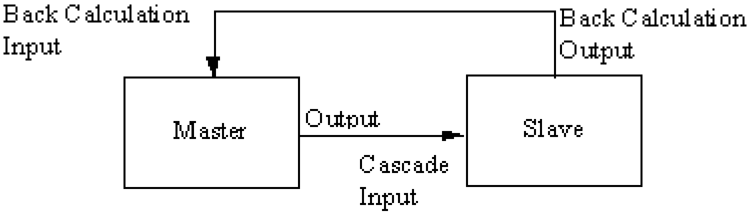
It is possible for the user to configure SHED_OPT so that it calls for a target mode that is not permitted. When doing this, the mode logic does the following rules as applied by the remote logic:

- Shed logic never results in a non-permitted target mode.
- Shed logic never attempts to attain an actual mode of Auto or Cas if that mode is not permitted.

Cascade Basics

Cascades in FOUNDATION Fieldbus follow a strict if simple set of rules. A cascade is a two (2) way communication. The driving or master block provides an output that is used as the cascade input by a lower block commonly referred to as the slave. The slave provides an output that informs the master as to when its output is being accepted and the limit conditions that exist below it; this output is the back calculation output. The master reads this input through a back calculation input. (Refer to Advanced Topics - BKCAL Communications for more information.) For each cascade mode in a block, there is at least one cascade input and one back calculation output. For each block that is defined as control block, there is at least one output and one back calculation input.

When the target mode is changed to a cascade mode in the slave, the back calculation output is set with a substatus of Initialization Requested (IR). Upon seeing IR at the back calculation input, the master sets Initialization Acknowledge (IA) in its output substatus. The combination of IR in its back calculation output and IA in its cascade input is the trigger for the slave block to change the actual mode to the cascade. This logic is generally applicable to all cascade inputs and outputs and cascade modes (Cas, RCas, ROut). (See below for the exception.) The following figure and table illustrate the initiation and completion of a cascade with a master target mode of Auto and a slave initial mode of Auto. In fact, the initial slave mode could be any of the operating modes Man, Auto, RCas, ROut. The master target mode could be Man, Auto, RCas, ROut, Cas and, as long as the actual mode was not being forced to LO, the logic completes, as shown in the following figure.



Cascade Configuration

Example Cascade with Auto as Target Mode

Substatus			**Mode**	
Condition	Slave BKCAL Output	Master OUT	Slave	Master
Cascade Is Not Possible	Not Invited	Anything	Not Cas	Target: Auto Actual: IMan
Initialization Requested by Slave	Initialization Requested	Anything but IA	Target: Cas Actual: Auto	Target: Auto Actual: IMan
Master Sees Initialization Request	Initialization Requested	Initialization Acknowledge	Target: Cas Actual: Auto	Target: Auto Actual: IMan

Substatus			Mode	
Slave Sees Initialization Acknowledge	Drops Initialization Requested and Changes to Normal	Initialization Acknowledge	Target: Cas Actual: Cas	Target: Auto Actual: IMan
Master Sees Normal	Normal	Drops Initialization Acknowledge and Changes To Normal	Target: Cas Actual: Cas	Target: Auto Actual: Auto, assuming no other forcing condition.

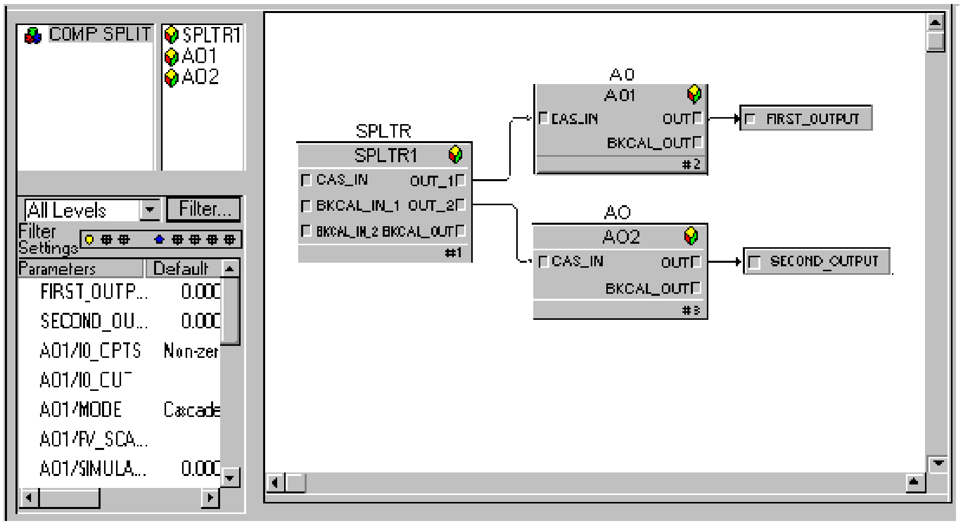
The exception to the cascade handshaking behavior described above occurs when a CAS_IN or CAS_IN_D parameter has a NonCascade substatus. This is the case when there is no master control block providing a GoodCascade status to CAS_IN, but there is a parameter or calculation block providing a GoodNonCascade status to CAS_IN. In such cases the receiving block does not need to see an InitializationAcknowledge (IA) on CAS_IN. If the target mode is Cas, the actual mode will change to Cas immediately if the status on CAS_IN is GoodNonCascade (and nothing else is preventing the actual mode from climbing).

Function Block Composites

A function block composite is a group of function blocks that works together. The function block composite holds an algorithm, or a portion of an algorithm, that you want to use again. Function block composites are similar to a subroutine in a software program that you call and use in multiple locations.

You build the function block composite once, store it in your Function Block Library, and reuse it as necessary for different loops and applications throughout your process or for other processes.

For an example, suppose you want to define an algorithm for a simple loop using function blocks. You could use a Splitter function block to generate two outputs and send them to two different Analog Output function blocks that convert the values and send the signal to field devices through output parameters. The following figure shows the function block composite diagram for this example.



Parameters	Default
FIRST_OUTP...	0.000
SECOND_OU...	0.000
AO1/IO_CPTS	Non-zer
AO1/IO_CU	
AO1/MODE	Cascade
AO1/PV_SCA...	
AO1/SIMULA...	0.000

Example Use of Splitter Function Block

For more information, refer to the Composites topic.

PT_COMP Composite Template

The PT_COMP composite template is used to compensate a measured flow based on measured pressure and/or temperature. The compensation can be used for gas or liquid streams where flow is measured using either a differential pressure flowmeter or a mass flowmeter. Flow compensation of a gas stream can be based on measured pressure, temperature or both. Compensation for liquid flow is based on measured temperature, using a linear relationship between temperature and density.

Inputs to the block are the measured flow, pressure, and temperature. The block output is the compensated flow. The status of the compensated flow is that of the measured flow. If the status of the measured pressure or temperature is other than Good, the reference (calibration) value is used in the calculation when applicable. This effectively disables compensation based on that input when the status is not Good. Status transitions are bumpless in downstream control blocks.

The compensation factor is calculated as follows:

Gas Stream

```
(( PRESSURE + ABS_PRESS_CF ) / ( REF_PRESS + ABS_PRESS_CF ))*(( REF_TEMP +
ABS_TEMP_CF ) / ( TEMPERATURE + ABS_TEMP_CF ))
```

Liquid Stream

((TEMPERATURE * DENSITY_M) + DENSITY_B) / REF_DENSITY

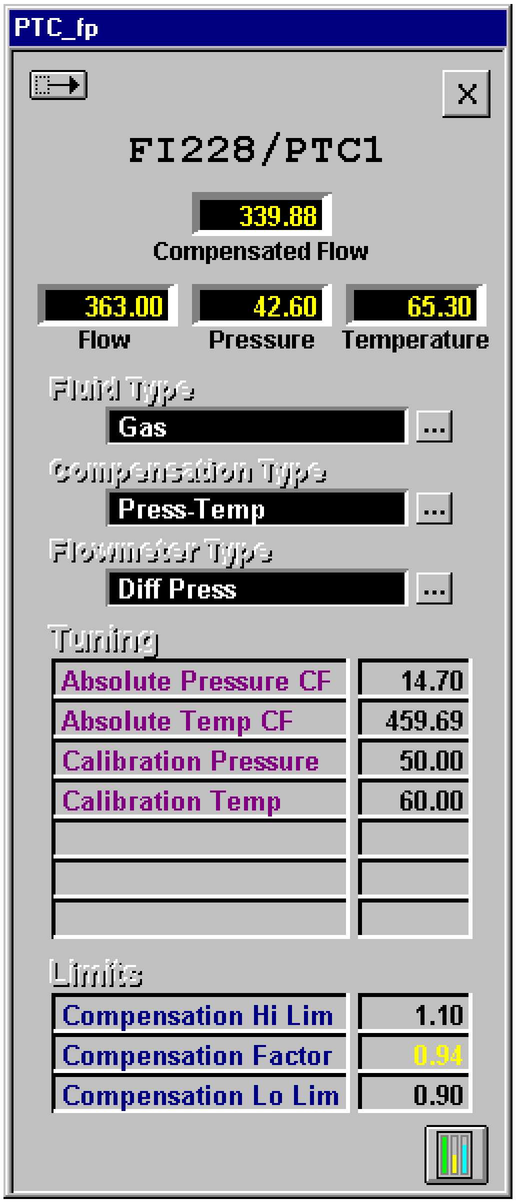
If the flowmeter is the differential pressure type, the compensation factor is the square root of the above equations.

You create a usage of this composite template by placing a custom block on the module's function block diagram. From the Special Items palette in Control Studio, drag the Custom Block icon onto the diagram. Give the block a usage name and select the block type as Linked Composite. Reference the existing object by browsing to the PT_COMP template.

This block can be configured using either Control Studio or DeltaV Explorer or from the faceplate in DeltaV Operate, provided the user has the Restricted Control security key. There is a dynamo for this composite function block in the frsFncblk Dynamo set in DeltaV Operate's configure mode.

PT_COMP Faceplate (PTC_FP)

The following figure shows the faceplate in DeltaV Operate configured for a gas stream. The value fields at the top of the faceplate indicate status by the background color (Good-Black, Uncertain-Orange, Bad-Red). The Pressure and Temperature value fields are visible based on the configured compensation type (COMP). The Compensation Factor under Limits is the actual ratio of the compensated flow to the measured flow. The other fields are configuration fields visible when the fluid type (FLUID) is Gas.



PTC_fp

FI228/PTC1

339.88
Compensated Flow

363.00	42.60	65.30
Flow	Pressure	Temperature

Fluid Type
Gas

Compensation Type
Press-Temp

Flowmeter Type
Diff Press

Tuning

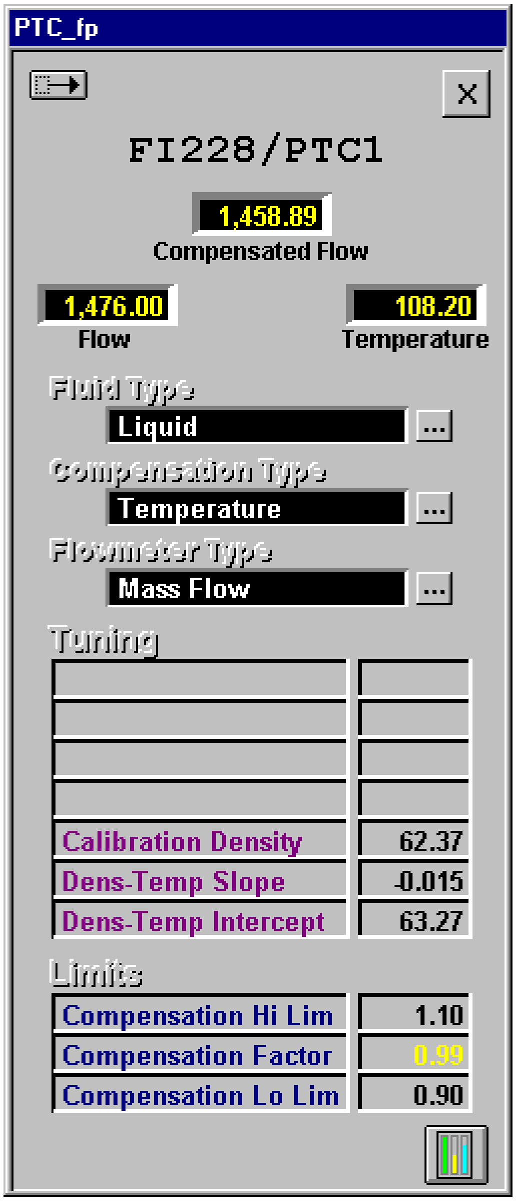
Absolute Pressure CF	14.70
Absolute Temp CF	459.69
Calibration Pressure	50.00
Calibration Temp	60.00

Limits

Compensation Hi Lim	1.10
Compensation Factor	0.94
Compensation Lo Lim	0.90

PT_COMP Faceplate with Fluid as Gas

The following figure shows the faceplate in DeltaV Operate with configuration fields visible when the fluid type is Liquid. Refer to the PT_COMP Quick Config Parameters topic.



FI228/PTC1

1,458.89
Compensated Flow

1,476.00
Flow

108.20
Temperature

Fluid Type
Liquid

Compensation Type
Temperature

Flowmeter Type
Mass Flow

Tuning

Calibration Density	62.37
Dens-Temp Slope	-0.015
Dens-Temp Intercept	63.27

Limits

Compensation Hi Lim	1.10
Compensation Factor	0.99
Compensation Lo Lim	0.90

PT_COMP Faceplate with Fluid as Liquid

Pressing the button at the bottom opens the module faceplate configured as a property of the module. As a default this is the generic module faceplate, MOD_FP.IAF. For more information, refer to the Pop-Up Displays for Function Blocks topic.

PT_COMP Quick Config Parameters

Quick Config Parameters for PT_COMP

Parameter	Description
ABS_PRESS_CF	The absolute pressure conversion factor. The value required to convert the gauge pressure to absolute pressure, in engineering units. The default value is 14.7.
ABS_TEMP_CF	The absolute temperature conversion factor. The value required to convert the temperature to degrees Rankine or Kelvin. If the temperature is measured in degF, use 459.69; if degC, use 273.16. The default value is 459.69.
COMP_HI_LIM	The compensation high limit. The default value is 1.1, which limits the maximum compensated flow to ten (10) percent higher than the measured flow.
COMP_LO_LIM	The compensation low limit. The default value is 0.9, which limits the minimum compensated flow to ten (10) percent lower than the measured flow.
COMP	The compensation type. Can be either Pressure, Temperature, or both. The default is both Press-Temp.
DENSITY_B	The density intercept value. Represents the intercept of the density curve. Used in calculating the actual density of the liquid when the FLUID parameter is set to Liquid. The default value is 63.27. The default value is for water with temperature measured in degrees Fahrenheit (degF).
DENSITY_M	The density slope value. Represents the slope of the density curve. Used in calculating the actual density of the liquid when the FLUID parameter is set to Liquid. The default value is -0.015. The default value is for water with temperature measured in degrees Fahrenheit (degF).
FLUID	The fluid type. Select either Gas or Liquid. The default value is Gas.
METER	The flowmeter type. The default value is Differential Pressure (Diff Press). Select the flowmeter type as either Differential Pressure or Mass Flow. The compensation factor is a square root function whenever the meter type is Diff Press. You must linearize the flow prior to this block if the meter type is differential pressure, for example, either in the field device or AI function block.
REF_DENSITY	The flowmeter calibration density. Enter the flowmeter calibration density. Temperature compensation of flow is done using a linear relationship between temperature and density. The actual density is calculated from temperature using the slope (DENSITY_M) and intercept DENSITY_B) of the linear relationship. The default values are for water with temperature measured in degrees Fahrenheit (degF). If the liquid is not water or units are not degF, enter values for the slope and intercept. The default value is 62.37.

Parameter	Description
REF_PRESS	The flowmeter calibration pressure. Enter the calibration pressure if the COMP parameter is set to Pressure or Press-Temp. This value is expected to be relative, for example, PSIG as opposed to PSIA. The calibration pressure should be in the same units as the measured pressure. The default value is 50.
REF_TEMP	The Flowmeter Calibration Temperature. Enter the calibration temperature if the COMP parameter is set to either Temperature or Press-Temp. This value is expected to be relative, for example, DEG C as opposed to DEG K. The flowmeter calibration temperature should be in the same units as the measured temperature. The default value is 60.

Function Block Status Information

Status values are generated by the following methods:

- Function block execution
- Status of I/O card and channel states (in I/O function blocks)
- Status values from upstream and/or downstream function blocks

A status value is expressed as: Quality State SubStatus LimitCondition (for example, Bad NonSpecific HighLimited).

Quality States

There are four quality states that define the status of an input, output, or contained parameter. They are:

- **Bad** – The value is not useful for control or calculation.
- **Uncertain** – The quality of the value is less than normal, but the value might be useful.
- **GoodNonCascade** – The quality of the value is good. Its source is not able to participate in cascade handshaking with the block receiving this status as an input, but the value can be used for control purposes without a cascade handshake. Refer to Cascade Basics for more information.
- **GoodCascade** – The value can be used in control and the source is able to participate in a cascade handshaking with the block receiving this status as an input. Refer to Advanced Topics - BKCAL Communications for more information.

Substatus Values

There are multiple substatus values that are defined for each quality state. The substatus values are described in the following list.

Bad substatus values

- **NonSpecific –** There is no specific reason why the value is bad. This substatus is used for signal status propagation.
- **ConfigErr –** Configuration Error. This substatus is set when the value is not useful because there is some other problem with the block. The reasons vary with each block and are noted in the individual block information.
- **NotConnected –** This substatus is set when this input is required to be connected and it is not connected.
- **DeviceFailure** – This substatus is set when the source of the value is affected by a device failure.
- **SensorFailure** – This substatus is set when the system can determine a sensor failure condition. The limit conditions (refer to the Limit Conditions section below) define which direction has been exceeded.
- **NoCommLUV** – No Communication, with last usable value (LUV). This substatus is set if this value was set by a communication that has failed.
- **NoCommNUV** – No Communication, with no usable value (NUV). This substatus is set when there has never been any communication with this value since it was last Out of Service.
- **OutOfService –** The value is not reliable because the block is not being evaluated and might be under construction during configuration. This substatus is set when the block mode is Out of Service (OOS).

Uncertain substatus values

- **NonSpecific –** There is no specific reason why the value is bad. This substatus is used for signal status propagation.
- **LUV** – Last Usable Value. The source that was writing this value has stopped writing the value. This happens when an input is disconnected during configuration.
- **Substitute** – This substatus is set when the value is written while the block is not Out of Service.
- **InitialValue** – This substatus is set when an input parameter value is written while the block is Out of Service.
- **SensorConversionNotAccurate –** This substatus is set when the value is at one of the sensor limits. The limit conditions (refer to the Limited Conditions section below) define which direction has been exceeded. In addition, this substatus is set when the device can determine that the sensor has reduced accuracy (such as a degraded analyzer). In this case, no limits are set.
- **EURangeViolation –** Engineering Unit Range Violation. This substatus is set when the value lies outside the range of values defined for this parameter. The limit conditions (refer to the Limited Conditions section below) define which direction has been exceeded.
- **SubNormal –** This substatus is set when a value derived from multiple values has less than the required number of *Good* sources.

GoodNonCascade *substatus values*

- **NonSpecific –** There is no specific reason why the value is bad. No error or special condition is associated with this value.
- **ActiveBlockAlarm** – This substatus is set when the value is good and the block has an active block alarm.
- **ActiveAdvisoryAlarm –** This substatus is set when the value is good and the block has an active alarm with a priority less than 8.
- **ActiveCriticalAlarm –** This substatus is set when the value is good and the block has an active alarm with a priority greater than or equal to 8.
- **UnacknowledgedBlockAlarm** - Set if the value is good and the block has an unacknowledged Block Alarm.
- **UnacknowledgedAdvisoryAlarm** - Set if the value is good and the block has an unacknowledged alarm with a priority less than 8.
- **UnacknowledgedCriticalAlarm** - Set if the value is good and the block has an unacknowledged alarm with a priority greater than 8.

GoodCascade substatus values

- **NonSpecific –** There is no specific reason why the value is bad. No error or special condition is associated with this value.
- **InitializationAcknowledge (IA) –** The value is an initialized value from a source through the CAS_IN parameter.
- **InitializationRequest (IR) –** The value is from a downstream block and causes re-initialization of the block. This substatus flag is not processed when the target mode is not Cascade (Cas).
- **NotInvited (NI) –** The value is from a block that does not have a target mode that uses this input (other than Local Override, and Not Selected).
- **NotSelected (NS) –** The value is from a Control Selector function block that has not selected the corresponding input. This substatus flag tells the upper block to limit in one direction (not to initialize).

- **DoNotSelect (DNS)** – The value is from a block that should not be selected by a Control Selector function block due to conditions in or above the block. Optionally, this status can be generated by a block when its actual mode is not Automatic (Auto) or Cascade (Cas).
- **LocalOverride (LO)** – The value is from a block that has been overriden locally. The failure of normal control must be propagated to a PID block for alarm and display purposes. This also implies Not Invited.
- **FaultStateActive (FSA)** – The value is from a block that has Fault State active. The failure of normal control must be propagated to a PID block for alarm and display purposes. This also implies Not Invited
- **InitiateFaultState (IFS)** – The value is from a block that wants its downstream output block (for example, AO) to go to Fault State.
- **FaultStateActive** – The output block has responded to a fault state condition and is in its defined fault state. When this status is on the output block's backcalculation output, the upstream control block treats it the same as Not Invited.
- **InitiateFaultState** – The primary output of a control block indicates that the downstream output block should go to its defined fault state. This is based on control options to initiate fault state if the status of the primary or cascade input is bad.

Limit Conditions

The following limit conditions are available in the status parameter. The four cases are mutually exclusive.

- **NotLimited** – The value is free to move.
- **LowLimited** – The value is from a block that cannot generate or use a lower value because it is limited in that direction, either internally or by the transducer.
- **HighLimited** – The value is from a block that cannot generate or use a higher value because it is limited in that direction, either internally or by the transducer.
- **Constant (high and low limited)** – The value cannot move. A constant cannot be limited in just one direction.

Refer to the Function Block Status Values topic for a complete list of status values.

Additional Status Information

Cascade loops have additional status states that concern loop initialization. The Fieldbus specification lists detailed information on this topic.

Specific status handling information for DeltaV function blocks is discussed in the Books Online information for each function block.

When a function block does not receive an input value as expected, the latest value is maintained and a stale data indicator is activated. When the input is reported as stale for more than a specified number of times, the status is set to Bad. In many function blocks, you can decide which (if any) action will be taken when the function block receives a stale data or Bad status indication.

Function Block Status Values

The following table lists all the function block status values.

Decimal Value	Status Value	Quality	Sub-Status	Limit
0	Bad NonSpecific NotLimited	Bad	NonSpecific	NotLimited
1	Bad NonSpecific LowLimited	Bad	NonSpecific	LowLimited
2	Bad NonSpecific HighLimited	Bad	NonSpecific	HighLimited
3	Bad NonSpecific Constant	Bad	NonSpecific	Constant
4	Bad ConfigErr NotLimited	Bad	ConfigErr	NotLimited
5	Bad ConfigErr LowLimited	Bad	ConfigErr	LowLimited
6	Bad ConfigErr HighLimited	Bad	ConfigErr	HighLimited
7	Bad ConfigErr Constant	Bad	ConfigErr	Constant
8	Bad NotConnected NotLimited	Bad	NotConnected	NotLimited
9	Bad NotConnected LowLimited	Bad	NotConnected	LowLimited
10	Bad NotConnected HighLimited	Bad	NotConnected	HighLimited
11	Bad NotConnected Constant	Bad	NotConnected	Constant
12	Bad DeviceFailure NotLimited	Bad	DeviceFailure	NotLimited
13	Bad DeviceFailure LowLimited	Bad	DeviceFailure	LowLimited
14	Bad DeviceFailure HighLimited	Bad	DeviceFailure	HighLimited
15	Bad DeviceFailure Constant	Bad	DeviceFailure	Constant
16	Bad SensorFailure NotLimited	Bad	SensorFailure	NotLimited

Decimal Value	Status Value	Quality	Sub-Status	Limit
17	Bad SensorFailure LowLimited	Bad	SensorFailure	LowLimited
18	Bad SensorFailure HighLimited	Bad	SensorFailure	HighLimited
19	Bad SensorFailure Constant	Bad	SensorFailure	Constant
20	Bad NoCommLUV NotLimited	Bad	NoCommLUV	NotLimited
21	Bad NoCommLUV LowLimited	Bad	NoCommLUV	LowLimited
22	Bad NoCommLUV HighLimited	Bad	NoCommLUV	HighLimited
23	Bad NoCommLUV Constant	Bad	NoCommLUV	Constant
24	Bad NoCommNUV NotLimited	Bad	NoCommNUV	NotLimited
25	Bad NoCommNUV LowLimited	Bad	NoCommNUV	LowLimited
26	Bad NoCommNUV HighLimited	Bad	NoCommNUV	HighLimited
27	Bad NoCommNUV Constant	Bad	NoCommNUV	Constant
28	Bad OutOfService NotLimited	Bad	OutOfService	NotLimited
29	Bad OutOfService LowLimited	Bad	OutOfService	LowLimited
30	Bad OutOfService HighLimited	Bad	OutOfService	HighLimited
31	Bad OutOfService Constant	Bad	OutOfService	Constant
64	Uncertain NonSpecific NotLimited	Uncertain	NonSpecific	NotLimited
65	Uncertain NonSpecific LowLimited	Uncertain	NonSpecific	LowLimited

Decimal Value	Status Value	Quality	Sub-Status	Limit
66	Uncertain NonSpecific HighLimited	Uncertain	NonSpecific	HighLimited
67	Uncertain NonSpecific Constant	Uncertain	NonSpecific	Constant
68	Uncertain LUV NotLimited	Uncertain	LUV	NotLimited
69	Uncertain LUV LowLimited	Uncertain	LUV	LowLimited
70	Uncertain LUV HighLimited	Uncertain	LUV	HighLimited
71	Uncertain LUV Constant	Uncertain	LUV	Constant
72	Uncertain Substitute NotLimited	Uncertain	Substitute	NotLimited
73	Uncertain Substitute LowLimited	Uncertain	Substitute	LowLimited
74	Uncertain Substitute HighLimited	Uncertain	Substitute	HighLimited
75	Uncertain Substitute Constant	Uncertain	Substitute	Constant
76	Uncertain InitialValue NotLimited	Uncertain	InitialValue	NotLimited
77	Uncertain InitialValue LowLimited	Uncertain	InitialValue	LowLimited
78	Uncertain InitialValue HighLimited	Uncertain	InitialValue	HighLimited
79	Uncertain InitialValue Constant	Uncertain	InitialValue	Constant

Decimal Value	Status Value	Quality	Sub-Status	Limit
80	Uncertain SensorConversionNotAccurate NotLimited	Uncertain	SensorConversionNotAccurate	NotLimited
81	Uncertain SensorConversionNotAccurate LowLimited	Uncertain	SensorConversionNotAccurate	LowLimited
82	Uncertain SensorConversionNotAccurate HighLimited	Uncertain	SensorConversionNotAccurate	HighLimited
83	Uncertain SensorConversionNotAccurate Constant	Uncertain	SensorConversionNotAccurate	Constant
84	Uncertain EURangeViolation NotLimited	Uncertain	EURangeViolation	NotLimited
85	Uncertain EURangeViolation LowLimited	Uncertain	EURangeViolation	LowLimited
86	Uncertain EURangeViolation HighLimited	Uncertain	EURangeViolation	HighLimited
87	Uncertain EURangeViolation Constant	Uncertain	EURangeViolation	Constant
88	Uncertain SubNormal NotLimited	Uncertain	SubNormal	NotLimited
89	Uncertain SubNormal LowLimited	Uncertain	SubNormal	LowLimited
90	Uncertain SubNormal HighLimited	Uncertain	SubNormal	HighLimited
91	Uncertain SubNormal Constant	Uncertain	SubNormal	Constant

Decimal Value	Status Value	Quality	Sub-Status	Limit
128	GoodNonCascade NonSpecific NotLimited	GoodNonCascade	NonSpecific	NotLimited
129	GoodNonCascade NonSpecific LowLimited	GoodNonCascade	NonSpecific	LowLimited
130	GoodNonCascade NonSpecific HighLimited	GoodNonCascade	NonSpecific	HighLimited
131	GoodNonCascade NonSpecific Constant	GoodNonCascade	NonSpecific	Constant
132	GoodNonCascade ActiveBlockAlarm NotLimited	GoodNonCascade	ActiveBlockAlarm	NotLimited
133	GoodNonCascade ActiveBlockAlarm LowLimited	GoodNonCascade	ActiveBlockAlarm	LowLimited
134	GoodNonCascade ActiveBlockAlarm HighLimited	GoodNonCascade	ActiveBlockAlarm	HighLimited
135	GoodNonCascade ActiveBlockAlarm Constant	GoodNonCascade	ActiveBlockAlarm	Constant
136	GoodNonCascade ActiveAdvisoryAlarm NotLimited	GoodNonCascade	ActiveAdvisoryAlarm	NotLimited
137	GoodNonCascade ActiveAdvisoryAlarm LowLimited	GoodNonCascade	ActiveAdvisoryAlarm	LowLimited
138	GoodNonCascade ActiveAdvisoryAlarm HighLimited	GoodNonCascade	ActiveAdvisoryAlarm	HighLimited
139	GoodNonCascade ActiveAdvisoryAlarm Constant	GoodNonCascade	ActiveAdvisoryAlarm	Constant
140	GoodNonCascade ActiveCriticalAlarm NotLimited	GoodNonCascade	ActiveCriticalAlarm	NotLimited

Decimal Value	Status Value	Quality	Sub-Status	Limit
141	GoodNonCascade ActiveCriticalAlarm LowLimited	GoodNonCascade	ActiveCriticalAlarm	LowLimited
142	GoodNonCascade ActiveCriticalAlarm HighLimited	GoodNonCascade	ActiveCriticalAlarm	HighLimited
143	GoodNonCascade ActiveCriticalAlarm Constant	GoodNonCascade	ActiveCriticalAlarm	Constant
144	GoodNonCascade UnacknowledgedBlockAlarm NotLimited	GoodNonCascade	UnacknowledgedBlockAlarm	NotLimited
145	GoodNonCascade UnacknowledgedBlockAlarm LowLimited	GoodNonCascade	UnacknowledgedBlockAlarm	LowLimited
146	GoodNonCascade UnacknowledgedBlockAlarm HighLimited	GoodNonCascade	UnacknowledgedBlockAlarm	HighLimited
147	GoodNonCascade UnacknowledgedBlockAlarm Constant	GoodNonCascade	UnacknowledgedBlockAlarm	Constant
148	GoodNonCascade UnacknowledgedAdvisoryAlarm NotLimited	GoodNonCascade	UnacknowledgedAdvisoryAlarm	NotLimited
149	GoodNonCascade UnacknowledgedAdvisoryAlarm LowLimited	GoodNonCascade	UnacknowledgedAdvisoryAlarm	LowLimited
150	GoodNonCascade UnacknowledgedAdvisoryAlarm HighLimited	GoodNonCascade	UnacknowledgedAdvisoryAlarm	HighLimited
151	GoodNonCascade UnacknowledgedAdvisoryAlarm Constant	GoodNonCascade	UnacknowledgedAdvisoryAlarm	Constant

Decimal Value	Status Value	Quality	Sub-Status	Limit
152	GoodNonCascade UnacknowledgedCriticalAlarm NotLimited	GoodNonCascade	UnacknowledgedCriticalAlarm	NotLimited
153	GoodNonCascade UnacknowledgedCriticalAlarm LowLimited	GoodNonCascade	UnacknowledgedCriticalAlarm	LowLimited
154	GoodNonCascade UnacknowledgedCriticalAlarm HighLimited	GoodNonCascade	UnacknowledgedCriticalAlarm	HighLimited
155	GoodNonCascade UnacknowledgedCriticalAlarm Constant	GoodNonCascade	UnacknowledgedCriticalAlarm	Constant
192	GoodCascade NonSpecific NotLimited	GoodCascade	NonSpecific	NotLimited
193	GoodCascade NonSpecific LowLimited	GoodCascade	NonSpecific	LowLimited
194	GoodCascade NonSpecific HighLimited	GoodCascade	NonSpecific	HighLimited
195	GoodCascade NonSpecific Constant	GoodCascade	NonSpecific	Constant
196	GoodCascade InitializationAcknowledge NotLimited	GoodCascade	InitializationAcknowledge	NotLimited
197	GoodCascade InitializationAcknowledge LowLimited	GoodCascade	InitializationAcknowledge	LowLimited
198	GoodCascade InitializationAcknowledge HighLimited	GoodCascade	InitializationAcknowledge	HighLimited
199	GoodCascade InitializationAcknowledge Constant	GoodCascade	InitializationAcknowledge	Constant

Decimal Value	Status Value	Quality	Sub-Status	Limit
200	GoodCascade InitilizationRequest NotLimited	GoodCascade	InitilizationRequest	NotLimited
201	GoodCascade InitilizationRequest LowLimited	GoodCascade	InitilizationRequest	LowLimited
202	GoodCascade InitilizationRequest HighLimited	GoodCascade	InitilizationRequest	HighLimited
203	GoodCascade InitilizationRequest Constant	GoodCascade	InitilizationRequest	Constant
204	GoodCascade NotInvited NotLimited	GoodCascade	NotInvited	NotLimited
205	GoodCascade NotInvited LowLimited	GoodCascade	NotInvited	LowLimited
206	GoodCascade NotInvited HighLimited	GoodCascade	NotInvited	HighLimited
207	GoodCascade NotInvited Constant	GoodCascade	NotInvited	Constant
208	GoodCascade NotSelected NotLimited	GoodCascade	NotSelected	NotLimited
209	GoodCascade NotSelected LowLimited	GoodCascade	NotSelected	LowLimited
210	GoodCascade NotSelected HighLimited	GoodCascade	NotSelected	HighLimited
211	GoodCascade NotSelected Constant	GoodCascade	NotSelected	Constant
212	GoodCascade DoNotSelect NotLimited	GoodCascade	DoNotSelect	NotLimited
213	GoodCascade DoNotSelect LowLimited	GoodCascade	DoNotSelect	LowLimited

Decimal Value	Status Value	Quality	Sub-Status	Limit
214	GoodCascade DoNotSelect HighLimited	GoodCascade	DoNotSelect	HighLimited
215	GoodCascade DoNotSelect Constant	GoodCascade	DoNotSelect	Constant
216	GoodCascade LocalOverride NotLimited	GoodCascade	LocalOverride	NotLimited
217	GoodCascade LocalOverride LowLimited	GoodCascade	LocalOverride	LowLimited
218	GoodCascade LocalOverride HighLimited	GoodCascade	LocalOverride	HighLimited
219	GoodCascade LocalOverride Constant	GoodCascade	LocalOverride	Constant
220	GoodCascade FaultStateActive NotLimited	GoodCascade	FaultStateActive	NotLimited
221	GoodCascade FaultStateActive LowLimited	GoodCascade	FaultStateActive	LowLimited
222	GoodCascade FaultStateActive HighLimited	GoodCascade	FaultStateActive	HighLimited
223	GoodCascade FaultStateActive Constant	GoodCascade	FaultStateActive	Constant
224	GoodCascade InitiateFaultState NotLimited	GoodCascade	InitiateFaultState	NotLimited
225	GoodCascade InitiateFaultState LowLimited	GoodCascade	InitiateFaultState	LowLimited
226	GoodCascade InitiateFaultState HighLimted	GoodCascade	InitiateFaultState	HighLimited
227	GoodCascade InitiateFaultState Constant	GoodCascade	InitiateFaultState	Constant

Function Block Alarm Detection

Some function blocks detect alarms based on the limit information you enter. The alarm detection capabilities vary with each function block. The Books Online information for each function block lists the configurable alarm limits for that block. The following limits are configurable in some function blocks:

High high (HI_HI_LIM)
High (HI_LIM)
Low low (LO_LO_LIM)
Low (LO_LIM)
Deviation high (DV_HI_LIM)
Deviation low (DV_LO_LIM)

You define the alarm priority, acknowledgment, and messaging information in Control Studio when you are configuring process control loops.

Some function blocks detect general block error information and set bitstring indicators to report the error. Block errors vary with each function block. The Books Online information for each function block lists the block errors for that block. The following block errors are supported in some function blocks:

Block configuration error – Cause varies by block

Input failure/process variable has *Bad status* – The source of the block's process variable is bad. Indicates a hardware failure, a non-existent Device Signal Tag (DST), or a Bad status on the SIMULATE parameter.

Local override – The block is in Local Override (LO) mode.

Other – Cause varies by block

Out of Service – The block is in Out of Service (OOS) mode.

Output failure – The output value is not valid.

Readback failed – The I/O readback failed.

Simulate active – Simulation is enabled and the block is using a simulated value in its execution.

You can use block error information to generate an alarm or send a message. Refer to the Alarms and Events topic for more information.

Fieldbus Function Block Information

Many of the function blocks in the DeltaV system are based on the standards set forth in the Fieldbus FOUNDATION Fieldbus Specifications. Many DeltaV function blocks are extended versions of the function blocks defined in the Fieldbus specifications or have additional capabilities you can use to perform advanced control or complex functions.

Function Blocks - Parameters

Parameters are data used in a function block to perform calculations and logic. Some parameters are defined and unchangeable for certain function blocks. Some parameters default to a most common value but can be modified by the user or another function block. Others must be set by the user before executing the function block. Parameters that can be changed are called writeable parameters.

Parameters can be described by the type of information they provide to the function block:

- Input parameters contain values from the operator, the field device, or other function blocks.
- Output parameters are the output values generated by the function block or set by the operator or another function block.
- Contained parameters are used only within the function block for calculation, logic, and status determination functions.
- Mode-controlled parameters vary with the mode of the block.

Parameters can be dynamic, static, or non-volatile, depending on how they are restored after power failure:

- Dynamic parameters are calculated by the block algorithm and do not need to be restored after power failure because the block recalculates the value.
- Static parameters have a specific value that must be restored by a device for use after power failure.
- Non-volatile parameters are written on a frequent basis and the last saved value (in most cases) is restored by a device after power failure.

Some parameters are extensible. That is, you can extend or increase the number of these parameters in a function block.

Some parameters are option bitstrings. They contain bit-encoded information that specify control strategy options, I/O value processing, status handling and processing, or the type of control logic used in the block.

Each parameter is transmitted in a certain data type format. This data can be transmitted between blocks for control, trending, alarming, and diagnostics.

The parameters visible in Control Studio may vary depending on level of configuration completed. For example, certain function blocks must be assigned to I/O before all the parameters are visible.

Creating Custom Engineering Unit Descriptors

When you configure the Units field of a scaling parameter such as PV_SCALE or OUT_SCALE, the DeltaV system has a standard list of engineering unit (EU) descriptors from which to choose. It is possible to add your own custom EU descriptors to this list from DeltaV Explorer. For instructions on creating custom engineering units, start DeltaV Explorer, select System Configuration | Setup and, in the right pane, select Engineering Units. Right click and select What's This? from the menu.

When exporting and importing the entire DeltaV system for an upgrade or reinstall, select the Include the data for a DeltaV software upgrade check box on the Export and Import dialogs to save and restore custom engineering unit descriptors.

Parameter Data Types

Parameter data can be transmitted between blocks and used for trending, alarming, diagnostics, and advanced control. Outputs from one function block can be brought into another function block as parameters for calculation or logic functions and sent to other function blocks or to the field.

Function block parameter data can be one of the following types:

Boolean – a logic value that is True (1) or False (0).
Boolean - Status – a True or False value with status indicator.

Discrete - Status – an unsigned 8-bit integer value with a status indicator.
Simulate Discrete – a discrete value that enables/disables simulation and a simulated discrete value and status.

Dynamic Reference – a type of external parameter used to define a path to a value selected at run time.

External Reference – a reference to a parameter outside the current module.

Floating Point – a number whose decimal point location is not fixed. This allows the calculation to consider significant digits information.
Floating Point Array – an aggregate of floating point values.
Floating Point with Status – a floating point value with a status indicator.
Simulate Float – a discrete value that enables/disables simulation and a simulated floating point value and status.

I/O Reference – a value that assigns channel names in an I/O device.

Integer – a whole number. A signed integer has a positive or negative sign associated with it; an unsigned integer does not. Integer values can be 8-, 16-, or 32-bit values.

Internal Reference – a reference to a parameter within the current module.

Mode – a collection of bitstrings that describe the target, actual, permitted, and normal modes of a block.

Named Set – a value from 0 to 255 that displays an assigned text message.

Option Bitstring – an unsigned 16-bit value that indicates the chosen options.

Scaling – a number that is used to convert a floating point value to the required engineering units.

You view the data type of each parameter by expanding the Parameter View window in Control Studio.

Because data of different types can be transmitted between function blocks, there are rules that govern data compatibility and conversions. The data conversions are performed automatically by the DeltaV software.

Data types that have a status associated with them are treated in one of three ways, depending on the conversion:

- When a data type with a status field is converted to another data type with a status field, the status field is copied from one to the other.
- When a data type with a status field is converted to a data type without a status field, the status field is lost.
- When a data type without a status field is converted to one with a status field, a *Good* status field is created.

The following table lists parameter data type compatibility information:

Function Block Parameter Compatibility

Parameter Data Type	Accepted Links	Comments
8-bit Signed Integer	8-bit Signed Integer	Bit copy full parameter.
	16-bit Signed Integer	Current value converted; no data loss.
	32 bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer with status	Current value converted; clamped to extremes of destination type.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.
16-bit Signed Integer	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Bit copy full parameter.
	32-bit Signed Integer	Current value converted; no data loss.

Parameter Data Type	Accepted Links	Comments
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer with status	Current value converted; clamped to extremes of destination type.
	Floating Point	Current value converted; no data loss.
	Floating Point with Status	Current value converted; no data loss.
	Discrete with Status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with Status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.
32-bit Signed Integer	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	32-bit Signed Integer	Bit copy full parameter.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer with status	Current value converted; clamped to extremes of destination type.

Parameter Data Type	Accepted Links	Comments
	Floating Point	Current value converted; precision loss.
	Floating Point with status	Current value converted; precision loss.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.
8-bit Unsigned Integer	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; no data loss.
	32-bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Bit copy full parameter.
	16-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Current value converted; no data loss.
	Discrete with Status	Current value converted; no data loss.

Parameter Data Type	Accepted Links	Comments
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; no data loss.
	Named Set	Current value converted; no data loss.
16-bit Unsigned Integer	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	32-bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Bit copy full parameter.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.

Parameter Data Type	Accepted Links	Comments
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; no data loss.
	Named Set	Current value converted; clamped to extremes of destination type.
32-bit Unsigned Integer	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	32-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Bit copy full parameter.
	32-bit Unsigned Integer with status	Current value converted; clamped to extremes of destination type.
	Floating Point	Current value converted; precision loss.
	Floating Point with status	Current value converted; precision loss.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.

Parameter Data Type	Accepted Links	Comments
32-bit Unsigned Integer with status	8-bit Signed Integer	Current value set to zero if source current value equals zero. Else, current value set to one.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	32-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer with status	Bit copy full parameter.
	Floating Point	Current value converted; precision loss.
	Floating Point with status	Current value converted; precision loss.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.
Boolean	8-bit Signed Integer	Current value converted; no data loss.
	16-bit Signed Integer	Current value converted; no data loss.
	32-bit Signed Integer	Current value converted; no data loss.

Parameter Data Type	Accepted Links	Comments
	8-bit Unsigned Integer	Current value converted; no data loss.
	16-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Current value converted; no data loss.
	Boolean	Bit copy full parameter.
	Boolean with status	Current value converted; no data loss.
	Option Bitstring	Current value converted; no data loss.
	Named Set	Current value converted; no data loss.
Boolean with status	8-bit Signed Integer	Current value converted; no data loss.
	16-bit Signed Integer	Current value converted; no data loss.
	32-bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Current value converted; no data loss.
	16-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Floating Point	Current value converted; no data loss.

Parameter Data Type	Accepted Links	Comments
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Current value converted; no data loss.
	Boolean	Current value converted; no data loss.
	Boolean with status	Bit copy full parameter.
	Option Bitstring	Current value converted; no data loss.
	Named Set	Current value converted; no data loss.
Floating Point	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	32-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer with status	Current value converted; clamped to extremes of destination type.
	Floating Point	Bit copy full parameter.
	Floating Point with status	Current value converted; clamped to extremes of destination type.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.

Parameter Data Type	Accepted Links	Comments
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.
Floating Point with status	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	32-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	32-bit Unsigned Integer with status	Current value converted; clamped to extremes of destination type.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Bit copy full parameter.
	Discrete with status	Current value converted; clamped to extremes of destination type.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; clamped to extremes of destination type.
	Named Set	Current value converted; clamped to extremes of destination type.
Floating Point Array	Floating Point Array	Compatible only when dimensions match exactly. Bit copy full parameter.

Parameter Data Type	Accepted Links	Comments
Simulate Float	None	
Discrete with status	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; no data loss.
	32-bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Current value converted; no data loss.
	16-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Bit copy full parameter.
	Mode	Current value converted into Target Mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; no data loss.
	Named Set	Current value converted; no data loss.
Simulate Discrete	None	
Option Bitstring	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; clamped to extremes of destination type.

Parameter Data Type	Accepted Links	Comments
	32-bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Current value converted; clamped to extremes of destination type.
	16-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Floating Point	Current value converted; no data loss.
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Current value converted; no data loss.
	Mode	Current value converted into target mode.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Bit copy full parameter.
	Named Set	Current value converted; clamped to extremes of destination type.
Scaling	Scaling	Bit copy full parameter.
Mode	8-bit Signed Integer	Actual mode converted into current value.
	16-bit Signed Integer	Actual mode converted into current value.
	32-bit Signed Integer	Actual mode converted into current value.
	8-bit Unsigned Integer	Actual mode converted into current value.
	16-bit Unsigned Integer	Actual mode converted into current value.

Parameter Data Type	Accepted Links	Comments
	32-bit Unsigned Integer	Actual mode converted into current value.
	32-bit Unsigned Integer with status	Actual mode converted into current value.
	Floating Point	Actual mode converted into current value.
	Floating Point with status	Actual mode converted into current value.
	Discrete with status	Actual mode converted into current value.
	Mode	Bit copy full parameter.
	Option Bitstring	Actual mode converted into current value.
	Named Set	Actual mode converted into current value.
Access Permissions	None	
I/O Reference	None	
Named Set	8-bit Signed Integer	Current value converted; clamped to extremes of destination type.
	16-bit Signed Integer	Current value converted; no data loss.
	32-bit Signed Integer	Current value converted; no data loss.
	8-bit Unsigned Integer	Current value converted; no data loss.
	16-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer	Current value converted; no data loss.
	32-bit Unsigned Integer with status	Current value converted; no data loss.
	Boolean	Current value set to zero if source current value equals zero. Else, current value set to one.
	Floating Point	Current value converted; no data loss.

Parameter Data Type	Accepted Links	Comments
	Floating Point with status	Current value converted; no data loss.
	Discrete with status	Current value converted; no data loss.
	Mode	Current value converted into target mode.
	Boolean with status	Current value set to zero if source current value equals zero. Else, current value set to one.
	Option Bitstring	Current value converted; no data loss.
	Named Set	Bit copy plus pointer duplication. Set strings must match.

For more detailed information about parameter data types, refer to the following topics:

- Floating Point Parameter
- Dynamic Reference Parameter
- External Reference Parameter
- Internal Reference Parameter

Parameter Filtering

Some function blocks have a large number of parameters. To help you quickly access the parameter information you need, the parameters are filtered. The Parameter View window in Control Studio shows the filtering that is currently applied to the parameters. When you want to change the system parameter filtering, use the DeltaV Explorer.

Standard system parameters have been put into filtered subsets to show you the parameters that are likely to be most meaningful to you for different tasks. These categories are not hierarchical or mutually exclusive. The system parameter subsets are:

Common Configurable – the parameters most commonly used during configuration.

Advanced Configurable – the parameters less commonly used during configuration or those used to configure advanced functions.

I/O References – the parameters that reference I/O signals from field devices.

Connectors – the parameters that reference the wired inputs and outputs of a function block or module.

Online Parameters – the parameters used for debugging an algorithm or for operating the process.

You can choose the subset(s) you want to view by clicking the filter buttons corresponding to the subsets you want to display. Turning on all five system parameter subsets ensures that you see all the system parameters associated with the block. Some parameters are in more than one system subset, but they are displayed only once in the parameter list regardless of the subsets you choose to view.

During configuration, you can select or define a grouping of all available parameters that best meets your needs. You do this by setting up the user-defined filter groups:

- Select the desired parameter and select Properties | Filter....
- Assign the parameter to one or more of the user groups.

The user parameter groups are:

- **Quick Config** – a group of parameters often used for the configuration of new modules. This grouping has a default list of parameters, but you can change them according to your needs. The parameters in this list have default values, which can help you configure a module quickly. However, you should examine each default value and modify the values that are not correct for your application.
- **User-defined 1** – a group of parameters you define for your process
- **User-defined 2** – a group of parameters you define for your process

You create custom parameter filtering when you configure a custom block. In a custom block, the default action assigns user-defined parameters to the Common Configurable and Online system subsets. However, you can assign the parameters to any system subset and user group you prefer.

Floating Point Parameter

A floating point parameter is a number that does not have a fixed decimal point location. The DeltaV system uses a standard IEEE, single precision, 32-bit floating point format. This format includes an overall sign bit, a signed exponent portion, and a mantissa portion. The exponent portion is applied to the base 2 and uses eight bits (seven bits and a sign bit). The mantissa, which is 23 bits, graduates the range determined by the exponent and establishes the resolution of the floating point number. The overall sign bit makes up the 32 bits.

From the above, it is evident that floating point numbers are represented by a finite number of bits. This implies that there is a minimum/maximum size for these numbers and that the number of significant digits that can be resolved is limited. The larger the number being stored, the poorer the resolution. In other words, the result of a mathematical operation involving floating point numbers would not be completely precise if one of the numbers was significantly larger than the other.

The 8-bit exponent allows a number between -2^128 and +2^128 (-3.4E10^38 and +3.4E10^38) to be represented. The mantissa determines the resolution of the number. With a mantissa of 23 bits, the resolution is 1/8,388,608 (1/(2^23)). This translates to seven significant decimal digits.

Floating point numbers in the DeltaV system have the following limits:

Maximum negative:	-3.4E10^38
Minimum negative:	-2.34E10^-39
Minimum positive:	+2.34E10^-39
Maximum positive:	-3.4E10^38

Mathematical operations resulting in numbers that exceed these limits will result in the limit value (for example, 3.4E10^38 + 100 = 3.4E10^38).

Dynamic Reference Parameter

A dynamic reference parameter is a variation of the external reference parameter that lets you define a path to a value that is selected at run time during execution of the algorithm. The selection is based on information not available at configuration (for example, an operator entry, a recipe parameter passed from batch control, or a run-time value of a control variable).

The fields for a dynamic reference parameter are described in the following table.

Dynamic Reference Parameter Fields

Name and Purpose	Type	Configurable	Readable	Writable
.$REF - Provides a means to change the parameter reference path or to read the path currently in use	String	No	Yes	Yes
.CST (Connection Status) - Provides a means to tell if the reference has been resolved (that is, the value has been found and can be read)	Integer	No	Yes -3 means 'external reference not resolved' -2 means 'parameter not configured' -1 means 'module not configured' 0 means 'good' 1 means 'not communicating'	No
.AWST (Asynchronous Write Status) - Provides a means to tell if the last attempt to write the referenced parameter was successful	Integer	No	Yes -4 means 'write rejected' -3 means 'external reference not resolved' -2 means 'parameter not configured' -1 means 'module not configured' 0 means 'success' 1 means 'not communicating' 2 means 'write pending'	No
.ST - The value of the ST field of the referenced parameter. Use this field to copy the ST value from the dynamically referenced field to a local parameter's ST field for use in subsequent ST sensitive calculations.	Integer	No	Yes	No

Note In the Expression Editor, the .CST field represents the communication status of the workstation and is parsed and validated as such in an expression. However, when the expression is downloaded, the .CST field is resolved as the reference parameter's communication status.

To allow flexibility when constructing or selecting dynamic reference paths, the use of strings is supported. A dynamic reference can be assigned a string constant (enclosed in quotation marks) or a string variable (enclosed in single quotes). Supported string functions, as described in the Strings topic, include the following:

- Numeric value to string conversion
- String to numeric value conversion
- **Equal** and **not equal** string comparison
- String concatenation
- A string selection function (SELSTR) that allows selection from five string constants or string parameters, based on an integer input to the string selection function

Dynamic references are established by assigning a parameter reference path string to the .$REF field of a dynamic reference parameter. For example, if a tank has two input valves, INLETA and INLETB, the value for the dynamic reference parameter, INLET.$REF, can be assigned the setpoint of INLET_A using a statement, such as:`'INLET.$REF' := "//INLETA/SP"`

Dynamic references should point to the parameter, not the field. You can now use 'INLET.CV' or 'INLET.ST', for example, in expressions to refer to the .CV and .ST fields of INLETA/SP. If you do not specify a field when using a reference in an expression, .CV is assumed.

Following is an example using a string variable to store the parameter reference path:`'STRINGVAR' := "INLETA/SP"`
`'INLET.$REF' := 'STRINGVAR'`

The use of the .CV field of named set parameters in string expressions is not prevented. However, it only functions if the states to be used in assignments are configured to be user selectable, and, therefore, the state names are held in controllers. The use of named sets to convert integer values to path strings is not recommended since named sets are system global and, for this purpose, they must be specific to the unit module or the unit class.

Note If you do not include a reference field when using a dynamic reference, .CV is assumed. To use another field you must include that field in your reference.

Dynamic references can be bound to module parameters or node parameters (including I/O parameters) in another node. You can also use existing (that is, non-dynamic) external reference parameters to place specific values into modules on this node.

Establishing a dynamic reference to a parameter in a unit module phase that is not loaded will not cause the phase to be loaded. Therefore, the phase attribute will not be read.

Writing a new string to the .$REF field immediately causes the values read from .CST and .AWST to go bad unless the connection is immediately established. It is recommended that the user test the value of these fields as needed.

Verifying Dynamic References

There are facilities in diagnostics to observe unresolved references. However, it is recommended that you establish and verify dynamic references in SFC pulse/assignment type actions. If a dynamic reference is used, you should make sure that it is bound (that is, a value is assigned) before continuing the step. This can be done using pulse action confirmation or a transition condition.

To establish and verify dynamic references, the SFC expression could be written as follows:

- The action expression resolves the dynamic reference path and assigns it to the .$REF field of the dynamic reference parameter at the module level or phase level.
- The confirmation expression tests for the value of the .CST field being equal to 0 (connected) or less than 0 (never going to connect).
- Subsequent expressions should test the .CST field of the individual dynamic reference parameters to guide the SFC execution if the algorithm is going to handle missing or IGNORE connections.

External Reference Parameter

An external reference allows you to refer to any input, output, or parameter available in the DeltaV system. External references are best configured using the parameter browser available in all DeltaV expression dialogs. The browser brings up a graphical list of areas, blocks, and parameters. By constructing a parameter path from the browser, you can avoid the potential of typographical errors and case sensitivity when referencing block parameters.

External reference parameters are denoted in expressions by surrounding them in single quotes (' '). For example, an external reference (EXT_REF1) could be constructed to point to the MODE parameter in a PID block (PID1) in another module (FIC_501). When EXT_REF1 is created, the path would be defined as:`FIC_501/PID1/MODE`

When referenced in a phase logic statement, the expression could be something like the following:

```
IF '/
EXT_REF1.ACTUAL' = MAN
THEN   OUT1 := 5.0
END_IF;
```

The fields for an external reference parameter are described in the following table.

External Reference Parameter Fields

Name and Purpose	Type	Configurable	Readable	Writable
.$REF - Provides a means to read the path currently in use	String	Yes	Yes	No
.CST (Connection Status) - Provides a means to tell if the reference has been resolved (that is, the value has been found and can be read). This is particularly useful if the external parameter is in another node.	Integer	No	Yes -3 means 'external reference not resolved' -2 means 'parameter not configured' -1 means 'module not configured' 0 means 'good' 1 means 'not communicating'	No
.AWST (Asynchronous Write Status) - Provides a means to tell if the last attempt to write the referenced parameter was successful	Integer	No	Yes -4 means 'write rejected' -3 means 'external reference not resolved' -2 means 'parameter not configured' -1 means 'module not configured' 0 means 'success' 1 means 'not communicating' 2 means 'write pending'	No
.ST - The value of the ST field of the referenced parameter. Use this field to copy the ST value from the externally referenced field to a local parameter's ST field for use in subsequent ST sensitive calculations. For more details, refer to the Function Block Status Values topic.	Integer	No	Yes	No

Note When using a reference field other than the .CV field, you will need to type the field name. The available fields are listed in the following table. For syntax rules regarding External references, refer to the External References topic.

External references can be bound to module parameters or node parameters (including I/O parameters) in another node.

Establishing an external reference to a parameter in a unit module phase that is not loaded will not cause the phase to be loaded. Therefore, the phase attribute will not be read.

Verifying External References

When referencing a parameter in another node, you can use the .CST field to verify communications with that parameter or node. The .CST field can be referenced in a CALC block or SFC expression. It is recommended that you verify external references to parameters in other nodes when using SFC pulse/assignment type actions. If such an external reference is used, you should make sure that it is bound (that is, a value is assigned) before continuing the step. This can be done using pulse action confirmation or a transition condition.

To verify such external references, the SFC expression could be written as follows:

- The confirmation expression tests for the value of the .CST field being equal to 0 (connected) or less than 0 (never going to connect).
- Subsequent expressions should test the .CST field of the individual dynamic reference parameters to guide the SFC execution if the algorithm is going to handle missing or IGNORE connections.

To verify such external references, the CALC expression could be written as follows:

- Set one of the OUT parameters equal to the .CST value of the remote node integrity or to a specific parameter such as RemoteNode1/OINTEG.CST or RemoteModule1/FB1/Parameter1.CST.
- Use the value of the CALC OUT parameter to adjust your logic based on the remote node or module being unavailable.
- Every time the module executes, the .CST value is updated with the current remote status.

Internal Reference Parameter

An internal reference allows you to refer to any input, output, or parameter available in the current module. Internal references are best configured using the parameter browser available in all DeltaV expression dialogs. The browser brings up a graphical list of valid blocks, and parameters. By constructing a parameter path from the browser, you can avoid the potential of typographical errors and case sensitivity when referencing block parameters.

The fields for an internal reference parameter are described in the following table.

Internal Reference Parameter Fields

Name and Purpose	Type	Configurable	Readable	Writable
.CST (Connection Status) - Provides a means to tell if the reference has been resolved (that is, the value has been found and can be read)	Integer	No	Yes -3 means 'external reference not resolved' -2 means 'parameter not configured' -1 means 'module not configured' 0 means 'good' 1 means 'not communicating'	No
.AWST (Asynchronous Write Status) - Provides a means to tell if the last attempt to write the referenced parameter was successful	Integer	No	Yes -4 means 'write rejected' -3 means 'external reference not resolved' -2 means 'parameter not configured' -1 means 'module not configured' 0 means 'success' 1 means 'not communicating' 2 means 'write pending'	No
.ST - For more details, refer to the Function Block Status Values topic.	Integer	No	Yes	No

Note When using a reference field other than the .CV field, you will need to type the field name. The available fields are listed in the following table. For syntax rules regarding internal references, refer to the Syntax Rules \| Internal Reference topic.

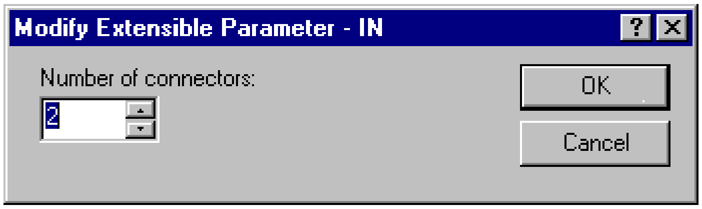
Verifying Internal References

It is recommended that you verify internal references that are referencing either dynamic referencing parameters or external reference parameters. Follow the appropriate instructions on how to verify those parameters.

Extensible Parameters

Some function blocks have extensible parameters. That is, you can extend or increase the number of this type of parameters. For example, you might need to extend the number of inputs on a function block so that you can input more values to the block without having to add more blocks to your diagram. The function blocks that are extensible are: Add, And, Boolean Fan Input, Boolean Fan Output, Calculation/Logic, Multiply, Multiplex, Or, and Signal Selector.

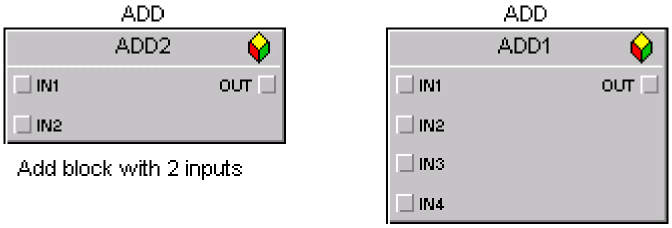
For example, the standard Add block from the palette has two inputs. When you want to add four values together, you extend the number of inputs by selecting the block, clicking the right mouse button, and changing the inputs number from 2 to 4. The following picture shows the dialog that you use to increase the inputs of an Add block:



Number of connectors: 2

Function Block Extensible Parameters Dialog Window

After you increase the number, the additional parameters show on the function block diagram and you can wire inputs to these parameters. Add blocks with two and four inputs are shown in the following figure.



Add block with 2 inputs

Add block with 4 inputs

ADD Function Blocks With Input Parameters Increased

Named Set Parameters

A named set is a group of system or user-defined, mutually exclusive descriptors that are assigned a numeric value. The descriptor is a text string that represents a single numeric value from 0 to 255. The named set is a group of such numerically valued descriptors that forms a list from which the operator or configuration engineer selects an item.

Named sets are used on any function block where it is important to define a value or set of values from which only one value at a time is used (for example, motor states, valve states, module states, and so on) or any item where discrete values are used.

Use the named set to select the value in Control Studio (when configuring the control logic for the block or module) or in DeltaV Operate (as part of the operator input).

Note The DeltaV software has default, preconfigured named sets that are accessed in DeltaV Explorer | Setup. User-defined named sets are configured in DeltaV Explorer. Named sets are case sensitive. They may be defined using upper case or lower case letters, but all future references to the state must be as originally defined (upper/lower case).

For example, when configuring control of a two-state motor, use the preconfigured named set mtr2-sp to define the values of the Set Point value. The named set mtr2-sp contains the following descriptors and values:

STOP = 0

START = 1

By choosing the named set mtr2-sp for the SP of the control block for the motor, you can design DeltaV Operate to allow the operator to choose the motor's state. This gives the operator the control of starting or stopping the motor when the need arises.

Selecting STOP for the SP sets the value to 0. Selecting START for the SP sets the value to 1. STOP and START are the only two values that are visible to the user and that the user can select for this named set.

The DeltaV software supports user-defined named sets. The user can configure named sets for any discrete value. For example, to indicate a level in a tank where the level indicator is a discrete device, create a named set (FILL_LEVEL) to indicate EMPTY, LOW, MED, and HIGH levels. The following is an example of the values for this named set:

EMPTY = 0

LOW = 1

MED = 2

HIGH = 3

Note The named set does not have to be visible to or selectable by the user. However, if you want the set to be selectable by the user, it must also be visible to the user. In addition, named set states have to be configured as user selectable for the state names to be available for use within the controller.

To configure named sets, start DeltaV Explorer and choose Setup | Named Set. Click the right mouse button and then click New Named Set from the context menu. Name the set (for example, FILL _LEVEL). Named sets are case sensitive. Select the new name set from the contents window (right window pane in Explorer). Click the right mouse button and then click Properties. Set the properties. Click OK. When configuring the control module for that tank and level indicator, use the named set FILL_LEVEL to indicate the current level of the tank.

Caution Avoid the following words in user-defined named sets. These names are used as system state names and are always interpreted with the following values, regardless of the user-defined value.

NO0
YES1
MAN8
AUTO16
CAS32, 48
RCas64
ROut128
OOS1
IMan2
LO4

Option Bitstring Parameters

Many function blocks have parameters that contain bit-encoded information to specify options that are available to you. There are several types of option bitstrings:

Control Options – allow you to select control strategy options.

I/O Options – allow you to select how the I/O signals are processed.

Status Options – allow you to select options for status handling and processing.

Integration Options – allow you to specify integration and reset behavior.

Device Options – allow you to select device controller logic options.

Algorithm Options – allow you to specify abort or no-abort on read errors.

Input Options – allow you to set the options for using IN and IN_LO when Uncertain. They also allow you to set the options for using IN_1, IN_2 and IN_3 when any are either Bad or Uncertain.

FSRI Add-On Options – allow you to select additional control and I/O options for the DeltaV blocks.

Interlock Option – Allows you to select the block's behavior when the status of INTERLOCK_D, PERMISSIVE_D, or SHUTDOWN_D is not Good.

Tracking Option – Allows you to select the tracking behavior when the status of the tracked variable is not Good.

Supported options vary with each function block. The following sections contain tables that list the blocks that implement each option. In addition, the Books Online information on each function block lists the supported options for that block.

Control Options

The control options parameter (CONTROL_OPTS) allows you to select control strategy options. You can set control options in Manual or Out of Service mode only. The following are the control options in the DeltaV system:

Use BKCAL_OUT With IN_1 – Normally, BKCAL_OUT is associated with initialization of an upstream block that is providing CAS_IN. If this option is set, then BKCAL_OUT is associated with the upstream block providing IN_1. This option is used with the Ratio and Bias/Gain block to determine the value and status that must be provided in BKCAL_OUT for proper initialization and handshaking.

No OUT Limits in Manual – Do not apply OUT_HI_LIM or OUT_LO_LIM when target and actual modes are Man.

Obey SP Limits if Cas or RCas – Normally the setpoint will not be restricted to the setpoint limits except when entered by a human interface device. However, if this option is selected, the setpoint will be restricted to the setpoint absolute limits in the Cas and RCas modes.

Act on IR – Used to back calculate for bumpless transfer.

In the Bias/Gain function block as well as the Ratio function block, if this option is true, when the block is in a non-automatic mode or transitions to an automatic mode, the SP will be back calculated for purposes of bumpless transfer. The back calculated SP will be limited to the range defined by the SP limits. Any difference between the value required for bumpless transfer will be added to OUT and then ramped out over BAL_TIME.

In the PID block, if this option is true and the STRUCTURE parameter is set to PD, then, when the block transitions to an automatic mode from a non-automatic mode the BIAS is back calculated for bumpless transfer.

Use PV for BKCAL_OUT – Normally, BKCAL_OUT contains the value of the working setpoint (SP_WRK). When necessary, use the process variable (PV) instead by turning on the Use PV for BKCAL_OUT control option. Note that even when this control option is turned on, it is enforced only when the block is in Cas mode. When the block is in a non-Cas mode, SP_WRK is used for BKCAL_OUT.

Track in Manual – Enables external tracking for target mode of MAN when Track Enable option has been selected. If Track In Manual is not selected then external tracking is prevented with a target mode of MAN, and, switching to a target mode of MAN will stop external tracking if external tracking had previously been initiated.

Track Enable – Enables external tracking function. If Track Enable option is selected, then, when TRK_IN_D is true, and, the target mode is not MAN or the Track In Manual option is selected, the block is placed in LO, the block output is set to the value of TRK_VAL.

Direct Acting – Defines the relationship between a change in PV and the corresponding change in output. When Direct Acting is enabled (True), an increase in PV results in an increase in the output.

SP Track Retained Target – Permits the setpoint to track the RCas or Cas parameter based on the retained target mode when the actual mode of the block is IMan, LO, Man, or ROut. SP Track Retained Target has precedence over the other SP-PV track options in the selection of the value to track when the actual mode is IMan, LO, Man, or ROut.

SP-PV Track in LO or IMan – Permits the SP to track the PV when the actual mode of the block is LO or IMan. SP-PV Track in Man takes precedence over SP-PV Track in LO or IMan. SP-PV Track in Man must be enabled in order for SP-PV Track in LO or IMan to track when Target mode is MAN.

SP-PV Track in ROut – Permits the setpoint to track the process variable when the actual mode of the block is ROut.

SP-PV Track in Man – Permits the SP to track the PV when the target mode of the block is Man.

Bypass Enable – This parameter, if true, allows BYPASS to be set. Some control algorithm applications cannot provide closed loop control if bypassed.

The following table lists the supported control options in specific function blocks.

Supported Control Options for Function Blocks

Function Block	Supported Control Options (CONTROL_OPTS)
Bias/Gain Function Block	No OUT limits in Manual Obey SP limits if Cas or Rcas Use BKCAL_OUT with IN_1 Act on IR Track in Manual Track Enable
FLC Function Block	Act on IR Use PV for BKCAL_OUT Track in Manual Track Enable Direct Acting SP-PV Track in LO or IMan SP-PV Track in Man Bypass Enable
Manual Loader Function Block	No OUT limits in Manual Track in Manual Track Enable

Function Block	Supported Control Options (CONTROL_OPTS)
PID Function Block	No OUT limits in Manual Obey SP limits if Cas or RCas Act on IR Use PV for BKCAL_OUT Track in Manual Track Enable Direct Acting SP Track Retained Target SP-PV Track in LO or IMan SP-PV Track in ROut SP-PV Track in Man Bypass Enable
Ratio Function Block	No OUT limits in Manual Obey SP limits if Cas or RCas Use BKCAL_OUT with IN_1 Act on IR Use PV for BKCAL_OUT Track in Manual Track Enable SP Track Retained Target SP-PV Track in Lo or Iman SP-PV Track in Man
Signal Characterizer Function Block	Bypass Enable

I/O Options

The I/O options parameter (IO_OPTS) allows you to select how the I/O signals are processed. You can set I/O options in Manual or Out of Service mode only. The following are the I/O options in the DeltaV system:

Low Cutoff – Changes any calculated output below this value to a zero output value. This is useful for zero-based measurement devices such as flowmeters.

Use PV for BKCAL_OUT – Changes the BKCAL_OUT value to the PV value. When the Use PV for BKCAL_OUT option is not enabled (False), the BKCAL_OUT value is the working setpoint value.

Target to Man if Fault State activated – Set the target mode to Man, thus losing the original target, if Fault State is activated. This latches an output block into the manual mode.

Use Fault State value on restart – Use the value of FSTATE_VAL(_D) if the device is restarted, otherwise use the non-volatile value. This does not act like Fault State, it only uses the value of FSTATE.

Fault State to value – The output action to take when fault occurs. (0: freeze, 1: go to preset value)

Increase to Close – Indicates whether or not the output value is inverted before it is communicated to the I/O channel.

SP-PV Track in LO or IMan – Permits the SP to track the PV when the actual mode of the block is LO or IMan. SP-PV Track in Man takes precedence over SP-PV Track in LO or IMan. SP-PV Track in Man must be enabled in order for SP-PV Track in LO or IMan to track when Target mode is MAN.

SP-PV Track in Man – Permits the SP to track the PV when the target mode of the block is Man.

Invert – Indicates whether the discrete input should be logically inverted before it is stored in the process variable. A discrete value of zero (0) will be considered to be a logical zero (0) and a non-zero discrete value will be considered to be a logical one (1). For example, if invert is selected, the logical NOT of a non-zero field value would result in a zero (0) discrete output, and the logical NOT of a zero field value would result in a discrete output value of one (1).

Note IMan is not possible in an I/O block.

The following table lists the supported I/O options for specific function blocks.

Supported I/O Options for Function Blocks

Function Block	Supported I/O Options (IO_OPTS)
Analog Input Function Block	Low Cutoff
Analog Output Function Block	Use PV for BKCAL_OUT Increase to Close SP_PV Track in Man
Discrete Input Function Block	Invert
Discrete Output Function Block	Use PV for BKCAL_OUT Tgt to Man if Fault St Act Use Fault st val on restart Fault State to value PS_PV Track in LO or IMan* SP_PV Track in Man Invert
FLC Function Block	Low cutoff Use PV for BKCAL_OUT Use Fault st val on restart Increase to close SP-PV Track in LO or IMan SP-PV Track in Man Invert
Multiple Discrete Input Function Block	Invert
Multiple Discrete Output Function Block	Use PV for BKCAL_OUT Tgt to Man if Fault State Active Use fault st val on restart Fault State to value SP-PV Track in Man SP-PV Track in LO or Iman Invert

Function Block	Supported I/O Options (IO_OPTS)
Multiplexed Analog Input	Low cutoff
PID Function Block	Low cutoff Use PV for BKCAL_OUT Use Fault st val on restart Increase to close SP-PV Track in LO or IMan SP-PV Track in Man Invert

* These parameters are only visible when the function block is extended to a Fieldbus device.

Status Options

The status options parameter (STATUS_OPTS) allows you to select options for status handling and processing. You can set status options in Manual or Out of Service mode only. The following are the status options in the DeltaV system:

Uncertain if Man mode – Set the output status of an input or calculation block to Uncertain if the actual mode of the block is Man.

BAD if Limited – Sets the output status to *Bad* if the sensor is at or beyond a high or low limit.

Uncertain if Limited – Set the output status of an input or calculation block to Uncertain if the measured or calculated value is limited.

Target to Manual if Bad IN – Sets the target mode to Man if the status of the IN parameter is BAD.

Propagate Fault Backward – If the status from the actuator is Bad - Device Failure or Fault State Active or Local Override is active, propagate this as Bad, Device Failure or Good Cascade, Fault State Active or Local Override to BKCAL_OUT respectively without generating an alarm. The use of these substatuses in BKCAL_OUT is determined by this option. Through this option, the user can determine whether alarming (sending of an alert) will be done by the block or propagated upstream for alarming.

Propagate Fault Forward – If the status from the sensor is Bad - Device Failure or Bad - Sensor Failure, propagate it to OUT without generating an alarm. The use of these substatuses in OUT is determined by this option. Through this option, the user can determine whether alarming (sending of an alert) will be done by the block or propagated downstream for alarming.

Use Uncertain as Good – If the status of the IN parameter is Uncertain, treat it as GOOD.

IFS if Bad CAS_IN – Set Initiate Fault State status in the OUT parameter if the status of the CAS_IN parameter is BAD.

IFS if Bad IN – Set Initiate Fault State status in the OUT parameter if the status of the IN parameter is BAD.

The following table lists the supported status options for specific function blocks.

Supported Status Options for Function Blocks

Function Block	**Supported Status Options (STATUS_OPTS)**
Analog Input Function Block	Uncertain if Man mode BAD if Limited Uncertain if Limited
Analog Output Function Block	Propagate Fault Backward IFS if Bad CAS_IN
Bias/Gain Function Block	Target to Man if Bad IN Use Uncertain as Good IFS if Bad CAS_IN IFS if Bad IN
Discrete Input Function Block	Uncertain if Man mode Propagate Fault Forward*
Discrete Output Function Block	Propagate Fault Backward*
FLC Function Block	Bad if Limited Target to Manual if Bad IN
Input Selector Function Block	Uncertain if Man mode Use Uncertain as Good
Multiplexed Analog Input Function Block	Uncertain if Man mode BAD if Limited Uncertain if Limited Propagate Fault Forward
PID Function Block	If block is not in a Fieldbus device: Bad if Limited Uncertain if Limited Target to Manual if Bad IN Use Uncertain as Good
If block is in a Fieldbus device: Target to Manual if Bad IN Use Uncertain as Good IFS if Bad CAS_IN IFS if Bad IN	Ratio Function Block
Target to Man if Bad IN Use Uncertain as Good IFS if Bad CAS_IN IFS if Bad IN	

* These parameters are only visible when the function block is extended to a Fieldbus device.

Integration Options

The integration options parameter (INTEG_OPTS) allows you to specify integration and reset behavior. You can set integration options in any mode. The following integration options are supported in the Integrator function block:

Carry – Determines what to do with the integration value after a reset condition is reached. When the Carry option is not enabled (False), the integrated value starts over (at zero or SP) when it is reset. When the Carry option is enabled (True), the excess values are carried over past the trip point into the next integration cycle as the initial value of the integrator.

Flow Reverse – Determines the net flow direction. When Flow Reverse is selected (True), negative increments are included in the integration.

Flow Forward – Determines the net flow direction. When Flow Forward is selected (True), positive increments are included in the integration.

When both Flow Forward and Flow Reverse are selected, positive and negative increments are included in the integration.

Device Options

The device options parameter (DEVICE_OPTS) allows you to select device controller logic options. You can set device options in any mode. The following device options are supported in the Device Control function block:

Interlock – When the Interlock option is enabled (True) and the INTERLOCK_D input is False, OUT_D is set to the Passive state and is held there until INTERLOCK_D is set to True.

SP Track – When the SP Track option is enabled (True), SP_D tracks OUT_D when the actual mode is Local Override (LO).

Warning Disabling this option can potentially start the equipment when unattended if the SP is left Active following a failed start due to an interlock.

Permissive – Allows OUT_D to transition from a Passive state to an Active state or from one Active state to another when SP_D requests. When the Permissive is enabled (True) and the PERMISSIVE_D input is set True, OUT_D is allowed to transition from the Passive state to an Active state or from one Active state to the other when SP_D requests.

Reset Required – When the Reset Required option is enabled (True) and OUT_D is forced to Passive because of an interlock, shutdown, or run confirmation failure, OUT_D remains at Passive until RESET_D is set to True.

Trip – When the Trip option is enabled (True) and an Active Confirm is lost for a time greater than TRIP_TIME, OUT_D is set to Passive. The SP_D value must return to Passive to clear the tripped condition. When the Reset Required option is enabled (True), RESET_D must also be set to True to clear the tripped condition.

Passive when Confirmed – When the Passive when Confirmed option is enabled (True) and the field device is confirmed to the desired Active state, OUT_D is changed to the Passive state.

Passive on Active Timeout – When the Passive on Active Tim-out option is enabled (True) and an Active state is not confirmed within the confirm time following a command to an Active state, OUT_D is changed to the Passive state.

Algorithm Options

The algorithm options parameter (ALGO_OPTS) allows you to specify abort or no-abort on read errors. The ALGO_OPTS parameter is supported in the Action, Calculation/Logic, and Condition function blocks.

AbortOnReadErrors – When selected, the expression algorithm aborts after encountering a read error on any parameter. A read error occurs, for example when a parameter in another controller cannot be read because the controllers are not communicating.

When the expression aborts, the following occurs:

Condition block – PRE_OUT_D retains its value from the previous scan

Action block – None of the assignment statements are executed

Calc/Logic block – None of the assignment statements are executed.

Input Options

The input options (INPUT_OPTS) allow you to set the options for using IN and IN_LO when Uncertain. They also allow you to set the options for using IN_1, IN_2 and IN_3 when any are either Bad or Uncertain. When a particular option is **not** selected and the status is Bad or Uncertain, the last usable value is used. A value is usable if the status is Good or if the status is not Good but the input option for that status (Bad or Uncertain) is selected to use it. To change the input options while using the on-line view, first change the mode to OOS. The following input options are supported in the Arithmetic function block:

IN_3 Use Bad – If selected, the IN_3 Use Bad option uses the value of IN_3, even if the status equals Bad.

IN_3 Use Uncertain – If selected, the IN_3 Use Uncertain option uses the value of IN_3, even if the status equals Uncertain.

IN_2 Use Bad – If selected, the IN_2 Use Bad option uses the value of IN_2, even if the status equals Bad.

IN_2 Use Uncertain – If selected, the IN_2 Use Uncertain option uses the value of IN_2, even if the status equals Uncertain.

IN_1 Use Bad – If selected, the IN_1 Use Bad option uses the value of IN_1, even if the status equals Bad.

IN_1 Use Uncertain – If selected, the IN_1 Use Uncertain option uses the value of IN_1, even if the status equals Uncertain.

IN_LO Use Uncertain – If selected, the IN_LO Use Uncertain option uses the value of IN_LO, even if the status equals Uncertain.

IN Use Uncertain – If selected, the IN Use Uncertain option uses the value of IN, even if the status equals Uncertain.

FRSI Add-On Options

There are two classes of FRSI add-on options, FRSIRB and FRSIPID. The FRSIRB options are add-on I/O options for the DeltaV Bias/Gain and Ratio blocks. The FRSIPID options are add-on control options for the DeltaV PID block.

ADD-ON I/O OPTIONS (FRSIRB_OPTS)

Treat IN1 as Wild

Causes cascade input on IN_1 to be interpreted as cascade closed, regardless of substatus on the function block.

ADD-ON CONTROL OPTIONS (FRSIPID_OPTS)

Dynamic Reset Limit

Is the reset term based upon the control parameter (BKCAL_OUT) of the downstream block. Prevents windup caused by limiting of the downstream block or from dynamic performance because of tuning. The downstream block must have Use PV for BKCAL_OUT set as an I/O option for Dynamic Reset Limit option to be used in the upstream block.

Use Nonlinear GainModification

When False, the gain modifier is 1.0. Otherwise, it is calculated.

Use Delayed OUT on Bad PV

When the status of PV becomes Bad and the mode transitions to Man because of the Bad PV status, the value of OUT is 5 scans previous.

Conditional Alarming Parameters

The following parameters are available when conditional alarming is enabled on a function block. Refer to the Conditional Alarming topic for more information.

Conditional Alarm Parameters

Parameter	Units	Description
HI_HI_ENAB HI_ENAB LO_ENAB LO_LO_ENAB DV_HI_ENAB DV_LO_ENAB	None	When false, the ACT parameter is forced to zero. You create the conditional event in another block that determines the value of this parameter by wiring the value to the *alarm*_ENAB input on this block.
HI_HI_DELAY_OFF HI_DELAY_OFF LO_DELAY_OFF LO_LO_DELAY_OFF DV_HI_DELAY_OFF DV_LO_DELAY_OFF	Seconds	When true, this parameter delays the time (in seconds) that it takes for corresponding _ACT parameter to clear (be false) after a primary alarm condition clears. Every time the conditional event becomes false, the timer resets.

Parameter	Units	Description
HI_HI_DELAY_ON HI_DELAY_ON LO_DELAY_ON LO_LO_DELAY_ON DV_HI_DELAY_ON DV_LO_DELAY_ON	Seconds	When true, this parameter delays the time (in seconds) that it takes for corresponding _ACT parameter to be true after a primary alarm condition occurs. Every time the conditional event becomes false, the timer resets.
HI_HI_ENAB_DELAY HI_ENAB_DELAY LO_ENAB_DELAY LO_LO_ENAB_DELAY DV_HI_ENAB_DELAY DV_LO_ENAB_DELAY	Seconds	When *alarm*_ENAB becomes non-zero, this parameter continues to force corresponding _ACT parameter to zero for the time specified (in seconds). The timer resets only when *alarm*_ENAB goes from zero to 1.
HI_HI_HYS HI_HYS LO_HYS LO_LO_HYS DV_HI_HYS DV_LO_HYS	Percent of EU range	This parameter is used as a deadband when resetting base alarm conditions for analog values. The block uses the value of *alarm*_HYS instead of the standard *alarm*_HYS, regardless of the value of the *alarm*_ENAB parameters.

I/O Blocks

This chapter contains information on input/output function blocks in the DeltaV system.

Alarm Detection Function Block

The Alarm Detection function block provides the ability to easily specify alarms on parameters that are obtained from I/O or from the results of other function block calculations. This function block takes user specified input and does high, low and deviation alarm detection. The parameters generated by the Alarm Detection function block can then be used to generate an alarm in DeltaV Operate.

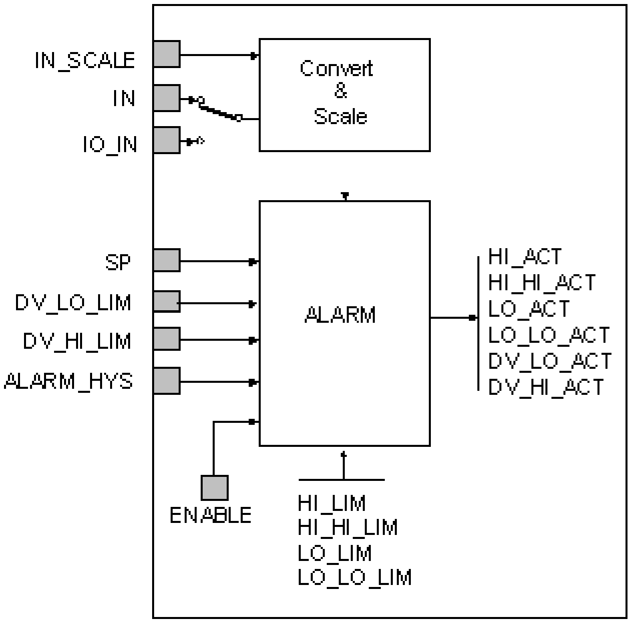
Modes are not supported in the Alarm Detection function block.

Alarm Detection Function Block

The IN is a floating point value and status.

Schematic Diagram - Alarm Detection Function Block

The following diagram shows the internal components of the Alarm Detection function block:



Alarm Detection Function Block Schematic Diagram

Block Execution - Alarm Detection Function Block

This function block detects six alarm conditions using the _LIM parameters (HI_LIM, HI_HI_LIM, LO_LIM, LO_LO_LIM, DV_LO_LIM, DV_HI_LIM). The result of the alarm detection uses the _ACT parameters that correspond to the appropriate _LIM parameter (HI_ACT, HI_HI_ACT, LO_ACT, LO_LO_ACT, DV_LO_ACT, DV_HI_ACT).

The value for all alarming is either wired to the IN parameter on the block, or referenced with IO_IN as an I/O path. Additionally, the SP value is used for deviation alarming.

When the ENABLE parameter is True(1), alarming is enabled. When ENABLE is False (0), alarming is disabled and all alarm XXX_ACT fields are cleared.

When the SCALE_ENABLE parameter is False, scaling is disabled. IN is expected in engineering units and written directly to PV.

When SCALE_ENABLE is True, IN is expected in percent. This percentage is applied to the range specified in IN_SCALE, which is in engineering units, to produce a PV in engineering units. The PV is then compared with the alarm limits, which are also in engineering units.

Status Handling - Alarm Detection Function Block

The following describes the Alarm Detection function block status handling:

- If IO_IN is not specified, IN is used unless its status is Bad or Not Connected.
- PV status shows Bad if IN is being used and IN status is Bad.
- PV status shows Bad if the I/O reference cannot be read or has a Bad status.
- Alarming is done regardless of the status of the input.

Parameters - Alarm Detection Function Block

The following table lists the system parameters for the Alarm Detection function block:

Alarm Detection Parameters

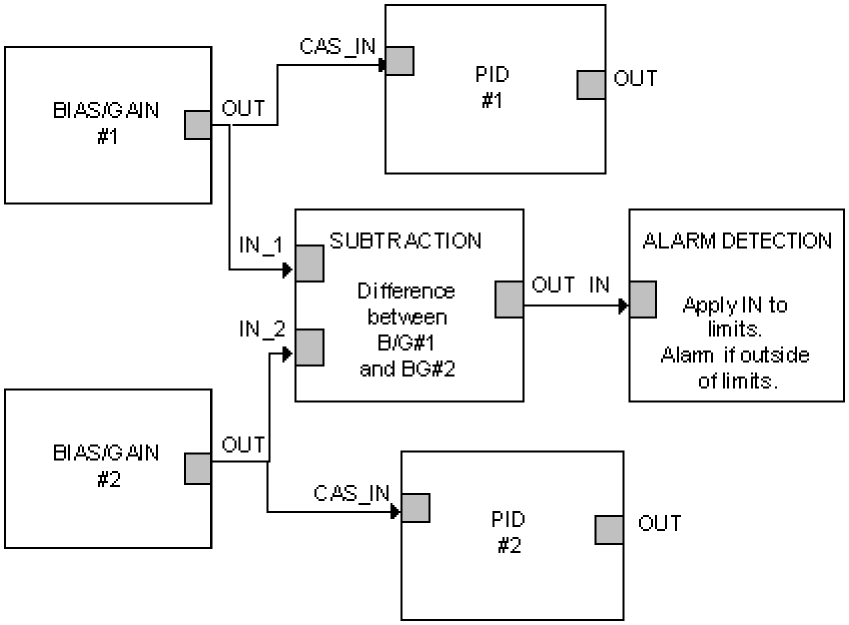
Parameter	Units	Description
ALARM_HYS	Percent or Percent of IN_SCALE	The amount the alarm value must return within the alarm limit before the associated active alarm condition clears. If SCALE_ENABLE is not set, the raw value will not be scaled. ALARM_HYS is limited to 50% of scale.
DEFAULT	EU of IN_SCALE	The value used for PV whenever IN status is bad. If SCALE_ENABLE is not set, the raw value will not be scaled.
DV_HI_ACT	None	The result of alarm detection associated with DV_HI_LIM. If DV_HI_ACT equals True, DV_HI_LIM has been exceeded.
DV_HI_LIM	EU of IN_SCALE	The amount by which PV can deviate above SP before the deviation high alarm is triggered. When this limit is exceeded, DV_HI_ACT is set to True. If SCALE_ENABLE is not set, the raw value will not be scaled.
DV_LO_ACT	None	The result of alarm detection associated with DV_LO_LIM. If DV_LO_ACT equals True, DV_LO_LIM has been exceeded.
DV_LO_LIM	EU of IN_SCALE	The amount by which PV can deviate below SP before the deviation low alarm is triggered. When this limit is exceeded, DV_LO_ACT is set to True. Note that DEV_LO_LIM is a negative number and is compared against (PV - SP). If SCALE_ENABLE is not set, the raw value will not be scaled.
ENABLE	None	The discrete input value and status that activates the alarm calculation.

Parameter	Units	Description
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.
HI_HI_LIM	EU of IN_SCALE	The setting for the alarm limit used to detect the high high alarm condition. If SCALE_ENABLE is not set, the raw value will not be scaled.
HI_LIM	EU of IN_SCALE	The setting for the alarm limit used to detect the high alarm condition. If SCALE_ENABLE is not set, the raw value will not be scaled.
IN	Determined by source or EU of PV_SCALE	The analog input value and status.
IN_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with IN or value from IO_IN.
IO_IN	None	Defines the input DST for the I/O channel used for the PV.
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.
LO_LIM	EU of IN_SCALE	The setting for the alarm limit used to detect the low alarm condition. If SCALE_ENABLE is not set, the raw value will not be scaled.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.
LO_LO_LIM	EU of IN_SCALE	The setting for the alarm limit used to detect the low low alarm condition. If SCALE_ENABLE is not set, the raw value will not be scaled.
PV	EU of IN_SCALE	The process variable used in block execution and alarm limit detection.
SCALE_ENABLE	None	Enables/disables scaling.
SP	EU of IN_SCALE	The block's setpoint value.

Application Information - Alarm Detection Function Block

You can use the Alarm Detection function block to check the outputs of two different blocks and set an alarm on a deviation range.

As an example, we have 2 motors that are operated at slightly different speeds. Therefore, two Bias/Gain blocks are needed, one for each motor. However, the difference between the two motors should never exceed a certain value. If it does, an alarm should trigger. Since the alarm is on the difference, a Subtraction block is used to calculate the difference between Motor 1 and Motor 2 and that output is evaluated by the Alarm Detection function block. When the OUT value from the Subtraction block is outside the limits set in the Alarm Detection block, an alarm triggers.



Use Example for Alarm Detection Function Block

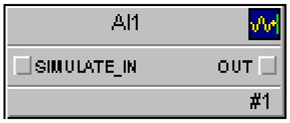
Analog Input (AI) Function Block

The Analog Input (AI) function block accesses a single analog measurement value and status from an I/O channel. You can configure the channel type for each I/O channel to be the transmitter's 4 to 20 mA signal or the digitally communicated primary or non-primary variable from a HART transmitter.

The AI function block supports block alarming, signal scaling, signal filtering, signal status calculation, mode control, and simulation.

In Automatic mode, the block's output parameter (OUT) reflects the process variable (PV) value and status. In Manual mode, OUT can be set manually.

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block through the SIMULATE_IN input.



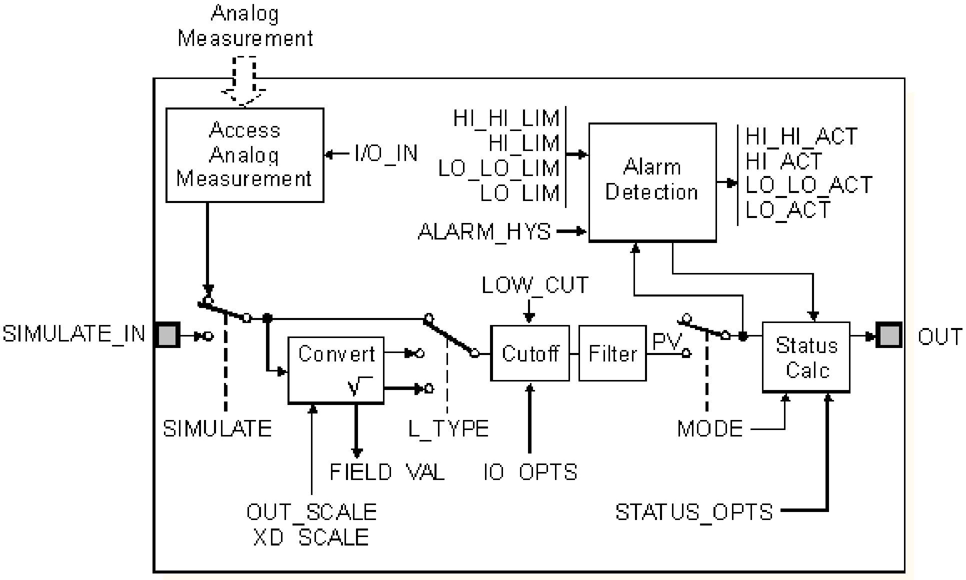
AI Function Block With Simulation Enabled

SIMULATE_IN is the simulated value from another block that is used by the Analog Input function block when simulation is enabled.

OUT is the block output value and status.

Schematic Diagram - Analog Input Function Block

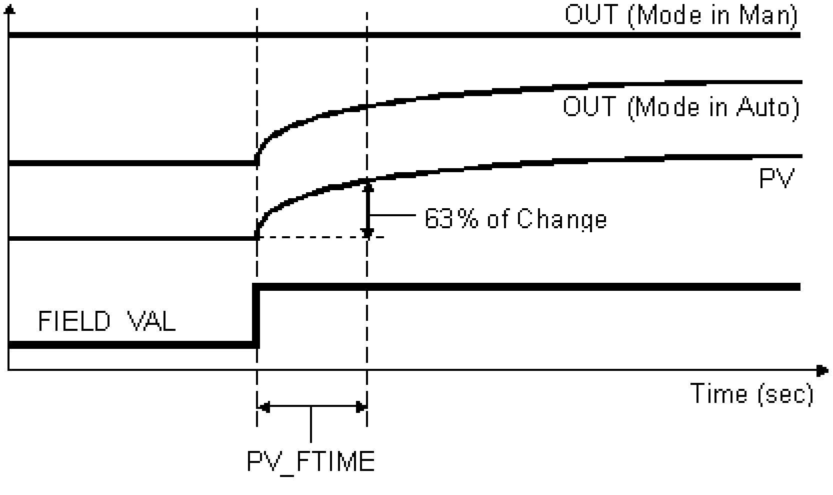
The following diagram shows the internal components of the Analog Input function block.



Analog Input Function Block Schematic Diagram

Block Execution - Analog Input Function Block

The following diagram shows the timed response of the Analog Input function block:



Analog Input Function Block Timing Diagram

You select the manner of processing the analog measurement value by configuring the I/O selection, signal conversion, and filtering parameters.

I/O Selection

When you configure the Analog Input function block, you select the I/O channel associated with an analog measurement by configuring the Device Signal Tag (DST) of the IO_IN parameter. You select the Device Tag and the parameter the AI block accesses on that channel.

When the channel type is configured as Analog Input Channel, the only selectable channel parameter for IO_IN is:

FIELD_VAL_PCT – the 4 to 20 mA signal, in percent of range. HART status is applied. You should configure XD_SCALE as 0 – 100%.

When the channel type is configured as HART Analog Input Channel, you can select one of the following parameters:

HART_FIELD_VAL – the 4 to 20 mA signal of the HART transmitter, in the range and engineering units defined by XD_SCALE or OUT_SCALE, depending on L_TYPE. HART Status is applied. The range and units are automatically sent to the transmitter to set its range and units. The specified units must be supported by the transmitter.

HART_PV – the Primary Variable of a HART transmitter, in engineering units. This value is read digitally.

HART_SV – the Secondary Variable of a HART transmitter, in engineering units. This value is read digitally.

HART_TV – the Tertiary Variable of a HART transmitter, in engineering units. This value is read digitally.

HART_FV – the Fourth Variable of a HART transmitter, in engineering units. This value is read digitally.

For more information on the DeltaV system's implementation of HART communications, refer to the HART Devices and the DeltaV System topic.

You can configure anti-aliasing filtering, NAMUR limit detection, and overrange/underrange detection for the channel parameters. For information on these capabilities, refer to the I/O Configuration topic.

Simulation

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block.

During configuration, decide whether you want the simulated value/status to be entered manually during operation or you will use a value/status from another block for the simulated value/status.

When the value is entered manually:

- The operator first enables simulation by selecting the SIMULATE parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field. In online operation, the operator can enter a simulated status value in the Simulate Status field.
- If SIMULATE_IN is not connected (status = *Bad: NotConnected*), the SIMULATE value is used.
- If SIMULATE_IN has any status except *Bad: NotConnected*, the SIMULATE_IN value is used. In this case, the SIMULATE_IN value always overrides the SIMULATE value.

Note Use the SIMULATE parameter to enter values manually and make sure SIMULATE_IN is not connected. If SIMULATE_IN is connected or a value is entered manually into SIMULATE_IN, the SIMULATE_IN value overrides the SIMULATE value.

When the value/status from another block is used:

- During configuration, connect SIMULATE_IN to the desired block output or parameter. Do not enter a value in the Simulate Value field of the SIMULATE_IN input; the block uses the connected value automatically.
- During operation, the operator enables simulation by selecting the SIMULATE parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.

Note Do not enter a value for the SIMULATE_IN parameter. If you do and the status of SIMULATE_IN is not *Bad: NotConnected*, the manually entered value for SIMULATE_IN overrides any value you enter in SIMULATE.

Signal Conversion

You choose direct, indirect, or indirect square root signal conversion with the linearization type parameter (L_TYPE). Refer to the HART Devices and the DeltaV System topic for information on signal conversion for HART devices.

Select one of the following signal conditioning options:

Direct signal conditioning – simply passes through the accessed channel input value (or the simulated value when simulation is enabled). For direct conversion, OUT_SCALE is written to XD_SCALE in controller-resident blocks.

Indirect signal conditioning – linearly converts the accessed channel input value (or the simulated value when simulation is enabled) from its specified range (XD_SCALE) to the range and units of the PV and OUT parameters (OUT_SCALE).

Indirect square root signal conditioning – converts the accessed channel input value (or the simulated value when simulation is enabled) from its specified range (XD_SCALE) by taking the square root of the value and scaling it to the range and units of the PV and OUT parameters (OUT_SCALE).

Direct Independent signal conditioning – simply passes through the accessed channel input values (or the simulated value when simulation is enabled). OUT_SCALE is independent of XD_SCALE.

You view the accessed value (in percent of XD_SCALE) through the FIELD_VAL parameter.

When the converted input value is below the limit specified by the LOW_CUT parameter and the Low Cutoff I/O option (IO_OPTS) is enabled (True), a value of zero is used for the converted value (PV). This option can be useful with zero-based measurement devices, such as flowmeters.

Note You can set the I/O option in Manual or Out of Service mode only.

Filtering

You apply filtering to the converted value (PV) by specifying the filter time constant (in seconds) in the PV_FTIME parameter. When you specify a value of zero, no filtering is applied.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Block configuration error – The block is wired to a HART channel and the HART device indicates that the function block's units are not compatible with the device. In this case, the Out of Service block error is also set.

Simulate active – Simulation is enabled and the block is using a simulated value in its execution.

Input failure/process variable has *Bad status* – The source of the block's process variable is bad. Indicates a hardware failure, a non-existent Device Signal Tag (DST), or a Bad status on the SIMULATE parameter.

Out of Service – The block is in Out of Service (OOS) mode.

Modes - Analog Input Function Block

The AI function block supports four modes:

Initialization Manual (IMan)

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of a block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Alarm Detection - Analog Input Function Block

Block alarm detection is based on the OUT value. You can configure the alarm limits of the following standard alarms:

- High (HI_LIM)
- High high (HI_HI_LIM)
- Low (LO_LIM)
- Low low (LO_LO_LIM)

NAMUR Alarming

If the high high and low low alarm limits are changed to a value outside the range defined in OUT_SCALE and the PV exceeds the alarm limits, the PV status high and low limits, respectively, are set. In this case, the associated active alarm parameters indicate that the measurement is overrange/underrange. If these alarm limits are set more than 3% outside the span of OUT, the active alarm condition is set after the overrange condition elapsed time exceeds four seconds.

Status Handling - Analog Input Function Block

Normally, the status of the measurement value provided by the analog input card reflects the operating condition of the I/O card, and processing condition.

If you configure the block to use a HART Analog Input Channel, you can select which conditions associated with a HART channel impact the status of the block. Refer to the HART Devices and the DeltaV System topic for information on these selections.

The channel input status to the analog input block is set high- or low-limited by the input card if the channel value exceeds the overrange and underrange limits and if the IO_IN parameter is selected as HART_FIELD_VAL or FIELD_VAL_PCT. you may change the default overrange and underrange limits independently on each input channel. Refer to the Overrange and Underrange Detection topic for more information. If the channel value is outside a range of -20.12 to 116.6%, the channel input status is set to BAD by the input card.

The (STATUS_OPTS) parameter allows you to choose how the PV and OUT status are determined. There are three AI block status options:

- Uncertain if Man mode -- Sets the OUT status to Uncertain if the block's mode is Manual
- Bad if Limited -- Sets the PV status to Bad if the channel input status is limited.
- Uncertain if Limited -- Sets the PV status to Uncertain if the channel input status is limited.

Note You can set the status option in Out of Service mode only.

The analog input block PV status normally reflects the channel input status. However, if the channel status is limited, then PV status is set to Uncertain limited or Bad Limited if you choose the status option Uncertain if Limited and Bad if limited respectively

In Auto mode, OUT normally reflects the value and status quality of the PV. However, if the PV status is Good but the PV value is more than -10% to 110% outside the span defined by PV_SCALE, then the OUT status is change to Uncertain.

In Man mode, the OUT status limit indication is set to constant and the OUT status is Good unless you have selected the Uncertain in Man mode status option.

In Out of Service mode, the OUT status is set to BAD.

NAMUR Limit Detection

If NAMUR limit detection is enabled for I/O channels (by setting the NAMUR_ENA parameter to True when configuring the AI card) and the IO_IN type is FIELD_VAL_PCT or HART_FIELD_VAL, the PV status of the input is set to *Bad* if the signal level is above 21 mA or below 3.6 mA for more than four seconds. The Bad status is cleared when the signal returns within these limits. You can use this feature when the transmitter is designed to flag a device failure by setting its current signal outside the normal 4-20 mA range.

Note The actual mode of a control block will automatically go to Man if the input has a Bad status. Thus, NAMUR limit detection should **not** be enabled if a measurement is used in control unless this behavior (control going to manual) is an appropriate action on limit detection.

Parameters - Analog Input Function Block

The following table lists the system parameters for the Analog Input function block:

Analog Input Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ACK_OPTION	None	Enables or disables automatic acknowledgment of fieldbus device alarms associated with Fuji, Endress&Hauser, and Honeywell devices. The default value is 0 (disabled). Set the value of this parameter to 65535 to enable the parameter before downloading the AI block. (0 = disabled (the default); 65535 =enabled).*
ALARM_HYS	Percent or Percent of PV_SCALE	The amount the alarm value must return within the alarm limit before the associated active alarm condition clears. ALARM_HYS is limited to 50% of scale.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Analog Input function block are Block configuration error, Simulate active, Input failure/process variable has *Bad* status, Output failure, and Out of Service.
CHANNEL	None	The number of the logical hardware channel that is connected to the I/O block. It defines the transducer to be used going to or from the physical world.*
FIELD_VAL	Percent	The value and status from the I/O card or from the simulated input when simulation is enabled.

Parameter	Units	Description
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.
HI_HI_LIM	EU of PV	The setting for the alarm limit used to detect the high high alarm condition.
HI_HI_PRI	None	The priority of the alarm detection associated with HI_HI_LIM and activated by HI_HI_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).*
HI_LIM	EU of PV	The setting for the alarm limit used to detect the high alarm condition.
HI_PRI	None	The priority of the alarm detection associated with HI_LIM and activated by HI_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).*
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 only if the following conditions are true: - The Write to Inspect Alarm context menu item has been selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition exists for Variability only if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
IO_IN	None	Defines the input DST for the I/O channel used for the PV.
IO_OPTS	None	I/O options. Allow you to select how the I/O signals are processed. The supported I/O option for the Analog Input function block is Low Cutoff.
L_TYPE	None	Linearization type. Determines whether the field value is used directly (Direct or Direct Independent), is converted linearly (Indirect), or is converted with the square root (Indirect Square Root).
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.

Parameter	Units	Description
LO_LIM	EU of PV	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.
LO_LO_LIM	EU of PV	The setting for the alarm limit used to detect the low low alarm condition.
LO_LO_PRI	None	The priority of the alarm detection associated with LO_LO_LIM and activated by LO_LO_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).*
LO_PRI	None	The priority of the alarm detection associated with LO_LIM and activated by LO_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).*
LOW_CUT	EU of PV	Activated when the Low Cutoff I/O option is enabled (True). When the converted measurement is below the LOW_CUT value, the PV is set to 0.0.
MODE	None	Parameter used to request and show the source of the output used by the block.
OUT	EU of OUT_SCALE	The primary value and status calculated by the block in Auto mode. OUT can be set manually in Man mode.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
PV	EU of OUT	The process variable used in block execution and alarm limit detection.
PV_FTIME	Seconds	The time constant of the first-order PV filter. It is the time required for a 63% change in the FIELD_VAL to be reflected in the PV.
SIMULATE	EU of XD_SCALE	Enables simulation and allows you to enter an input value and status. The SIMULATE value is used by the block only when SIMULATE_IN is not connected.

Parameter	Units	Description
SIMULATE_IN	EU of XD_SCALE	The input connector value and status used by the block instead of the analog measurement when simulation is enabled. If SIMULATE_IN is connected or has a manually entered value, SIMULATE_IN always overrides SIMULATE. **Note** When SIMULATE_IN is wired from an input source on the function block diagram, it always overrides a manually entered value.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*
STDEV_CAP	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds (reports in percent to Inspect)	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.
STDEV	EU of OUT_SCALE or EU of PV_SCALE	The standard deviation of PV. For analog control blocks in AUTO, mean is assumed to be the SP. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The supported status options for the Analog Input function block are: - Uncertain if Man mode - Bad if Limited - Uncertain if Limited

Parameter	Units	Description
SUBSTITUTE_IN	Determined by source or EU of PV_SCALE	If there is a problem communicating with the field device and SUBSTITUTE_IN has a non-Bad status, the controller passes the SUBSTITUTE_IN value through to the output parameter(s) of the shadow block.*
XD_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the channel input value.

* These parameters are only visible when the function block is extended to a Fieldbus device.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Analog Input Function Block

The method you use to configure an Analog Input function block and its associated input channel depends on whether the measurement is from a traditional 4 to 20 mA transmitter or a HART transmitter.

Traditional 4 to 20 mA Transmitter

When you are using a measurement from a traditional 4 to 20 mA transmitter, you configure the associated input channel and Analog Input function block as follows:

Channel Type: Analog Input Channel

Analog Input block IO_IN parameter: FIELD_VAL_PCT

Analog Input block L_TYPE parameter: Select Indirect when the measurement is linearly related to the 4 to 20 mA signal. Select Indirect Square Root when this is a flow measurement using differential pressure and when square root extraction is not performed in the transmitter.

Analog Input block XD_SCALE parameter: Set the range and engineering units to 0 – 100%.

Analog Input block OUT_SCALE parameter: Set the range and engineering units to the values that correspond to the transmitter's 4 to 20 mA signal.

HART Transmitter

Refer to the HART Devices and the DeltaV System topic for additional information. When you are using a measurement from a HART transmitter connected to an analog input channel, you configure the associated input channel and Analog Input function block as follows:

Channel Type: HART Analog Input Channel

Analog Input block IO_IN parameter: The analog input card passes the percent or engineering unit measurement of the primary variable (or the engineering unit value of a non-primary variable) based on the selection of IO_IN. The following table lists the IO_IN selections:

Analog Input Function Block IO_IN Parameter Selections for HART Transmitters

Variable	**Access**	**IO_IN Selection**	
		Engineering Units	Percent
Primary	4 to 20 mA	HART_FIELD_VAL	FIELD_VAL_ PCT
Primary	Digital	HART_PV	N/A
Secondary	Digital	HART_SV	N/A
Tertiary	Digital	HART_TV	N/A
4th	Digital	HART_FV	N/A

Note When a channel value is accessed digitally, the update rate of the analog input channel value depends on the number of channels on the associated I/O card that are configured as HART Analog Input channels. When all eight channels are configured as HART channels, each channel is updated every six seconds. When there is only one such channel on the card, the channel is updated approximately every 0.5 seconds.

Therefore, when you are using the primary measurement in closed loop control of a fast or moderately fast process, we recommend that you define IO_IN to access the 4 to 20 mA signal. You can use the digital value when you require the improved accuracy and range of the digital value for a monitoring application or for a control application involving a slow process.

Analog Input block L_TYPE parameter: Select Direct to use an I/O parameter that provides its signal in the engineering units that you want for the block output.

Select Indirect when you want to convert the I/O parameter value to engineering units based on the input and output ranges defined by XD_SCALE and OUT_SCALE.

Select Indirect Square Root when the block I/O parameter value represents a flow measurement made using differential pressure, and when square root extraction is not performed by the transmitter.

Select Direct Independent for RTD and Thermocouple inputs to allow OUT_SCALE to be set to a narrower range than XD_SCALE.

Analog Input block OUT_SCALE parameter: Set the desired ranges and engineering units for the PV and OUT parameters to correspond to the XD_SCALE range and the L_TYPE conversion.

Analog Input block XD_SCALE parameter: When the IO_IN channel type is FIELD_VALUE_PCT, configure XD_SCALE as 0 – 100%. When any other channel type is selected, configure XD_SCALE to match the range and units of the transmitter.

Note When you select HART_FIELD_VAL as the channel type, the range and units defined by the XD_SCALE or the OUT_SCALE parameter are written to the transmitter. This assumes that the specified engineering units are supported by the transmitter. If the transmitter units you specify are not supported by the transmitter, the channel status is set to *Bad*.

When the range or units of the transmitter's primary variable for your application is different than that defined in OUT_SCALE, set L_TYPE to Indirect and define the input and output ranges in XD_SCALE and OUT_SCALE.

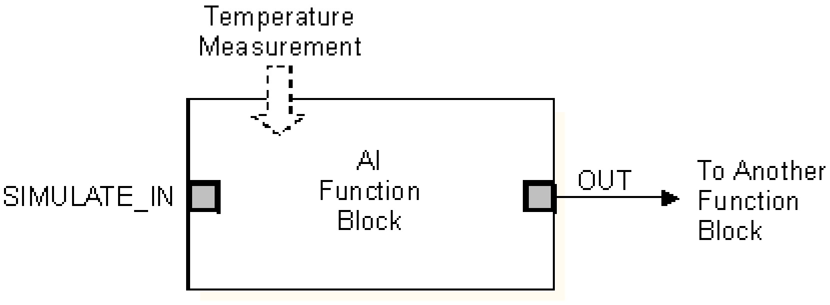
Application Example: Traditional Temperature Transmitter

Assume a temperature transmitter is calibrated for a range of 200 to 400°F and is used in a monitor (non-control) application in which accuracy is important. You configure the associated input channel and Analog Input function block as follows.

Analog Input Function Block Configuration Example for a Traditional Temperature Transmitter

Configuration Setting	Traditional 4 to 20 mA Transmitter	HART Transmitter
Channel Type	Analog Input	HART Analog Input
AI Block IO_IN Parameter	FIELD_VAL_PCT	HART_PV
L_TYPE	Indirect	Direct
XD_SCALE	0 to 100%	200 to 400°F
OUT_SCALE	200 to 400°F	200 to 400°F

The following figure is the function block diagram for this example.



Analog Input Function Block Diagram Example for a Traditional Temperature Transmitter

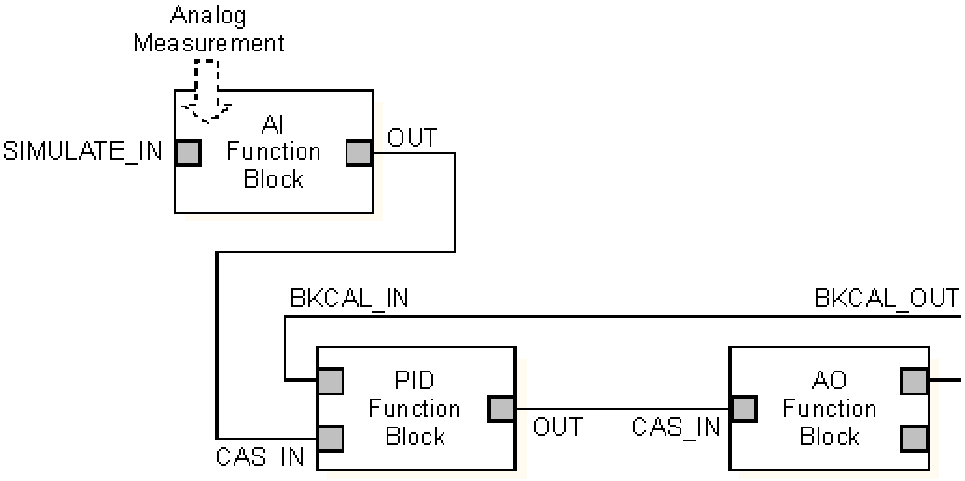
Application Example: Traditional Pressure Transmitter

Assume the level of a small open tank is to be measured using a pressure tap near the bottom of the tank. Based on the tap location and the density of the material in the tank, the pressure transmitter is calibrated at 0 to 200 in. H2O for a tank level of 0 to 10 ft. The level measurement is used to control the tank level. You configure the associated input channel and Analog Input function block as follows.

Analog Input Function Block Configuration Example for a Traditional Pressure Transmitter

Configuration Setting	Traditional 4 to 20 mA Transmitter	HART Transmitter
Channel Type	Analog Input	HART Analog Input
AI Block IO_IN Parameter	FIELD_VAL_PCT	HART_FIELD_VAL
L_TYPE	Indirect	Indirect
XD_SCALE	0 to 100%	0 to 200 in.
OUT_SCALE	0 to 10 ft.	0 to 10 ft.

The following figure is the function block diagram for this example.



Analog Input Function Block Diagram Example for a Traditional Pressure Transmitter

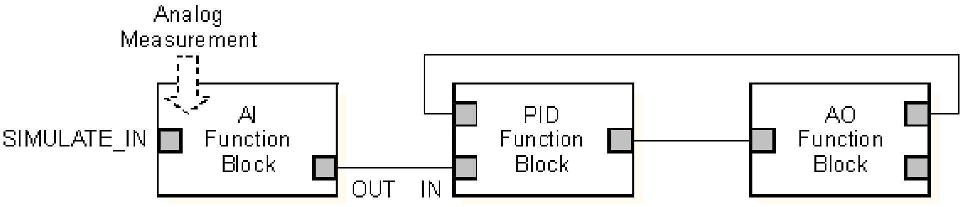
Application Example: Traditional Differential Pressure Transmitter

Assume the liquid flow in a line is to be measured using the differential pressure across an orifice plate in the line. Based on the orifice specification sheet, the differential pressure transmitter was calibrated for 0 to 20 in. H20 for a flow of 0 to 800 gal/min. The flow measurement is used in a flow control loop. If the transmitter was not set up to take the square root of the differential pressure, you configure the associated input channel and Analog Input function block as follows.

Analog Input Function Block Configuration Example for a Traditional Differential Pressure Transmitter

Configuration Setting	Traditional 4 to 20 mA Transmitter	HART Transmitter
Channel Type	Analog Input	HART Analog Input
AI Block IO_IN Parameter	FIELD_VAL_PCT	HART_FIELD_VAL
L_TYPE	Indirect Square Root	Indirect Square Root
XD_SCALE	0 to 100%	0 to 20 in.
OUT_SCALE	0 to 800 gal/min	0 to 800 gal/min

The following figure is the function block diagram for this example.



Analog Input Function Block Diagram Example for a Traditional Differential Pressure Transmitter

Advanced Information - Analog Input Function Block

The analog input card contains hardware filtering that limits the frequency of the input signal seen by the digital-to-analog (D/A) converter to about 3 Hz. In addition, you can define software filtering to be applied at the analog card when you configure the analog input channel properties (FILTER channel parameter). This feature prevents aliasing of the signal if the module execution rate is set slower than twice the highest frequency component of the input signal.

The default setting for the software filter is Filter Disabled. If an input signal is relatively free of process noise and is contained in a module executing at fast to moderate rates, you might not need to apply additional software filtering at the card.

If you choose to modify the filter, you see a selection of filter time constants in the Named State list. If you follow the module execution rate guideline (period of control) that is shown in parentheses next to the filter time constant you select, no aliasing occurs, even if the signal has significant noise.

Note If you modify the filter on an input channel, you might need to retune any control block that uses the channel for its controller variable.

Analog Output (AO) Function Block

The Analog Output (AO) function block assigns an output value to a field device through a specified I/O channel. The block supports mode control, signal status calculation, and simulation.

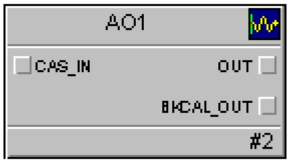
In Manual mode, the value of the output parameter (OUT) is set manually.

In Automatic mode, OUT is set automatically based on the value specified by the setpoint (SP) in engineering units and the I/O options parameter (IO_OPTS). In addition, you can limit the SP value and the rate at which a change in the SP is passed to OUT.

In Cascade mode, the cascade input connection (CAS_IN) is used to update the SP. The back calculation output (BKCAL_OUT) is wired to the back calculation input (BKCAL_IN) of the upstream block that provides CAS_IN. This provides bumpless transfer on mode changes and windup protection in the upstream block. The OUT parameter or an analog read-back value, such as valve position, is shown by the process value (PV) parameter in engineering units.

To support testing, you can enable simulation. This allows the channel feedback to be set manually.

There are no standard alarms in the Analog Output function block. Custom alarms are supported in this function block.



Analog Output Function Block Used in a Simulation

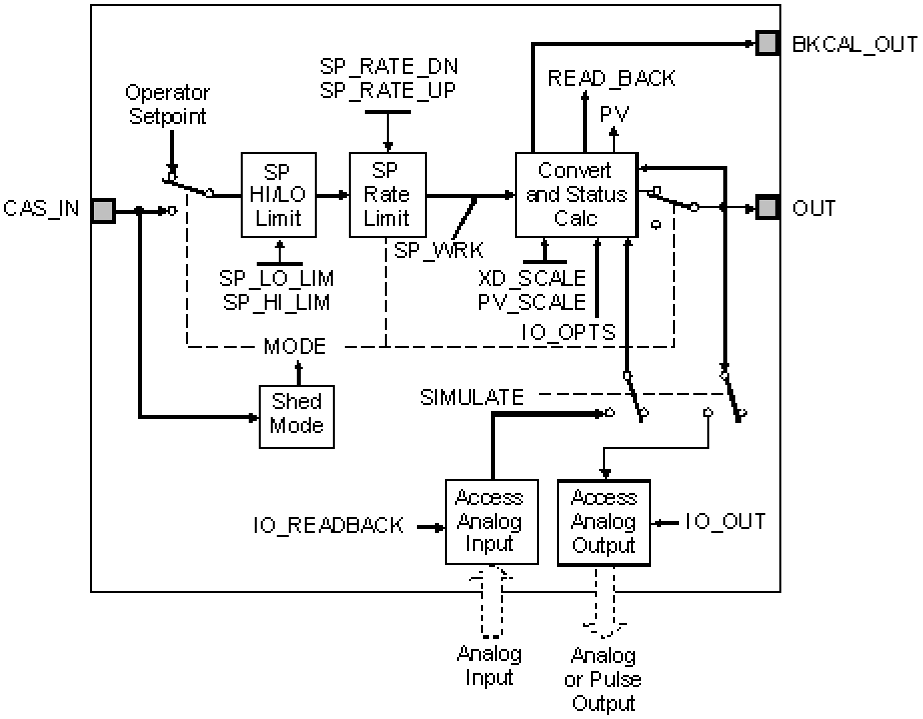
CAS_IN is the remote setpoint value from another function block.

BKCAL_OUT is the value and status required by the BKCAL_IN input of another block to prevent reset windup and to provide bumpless transfer to closed loop control.

OUT is the block output and status.

Schematic Diagram - Analog Output Function Block

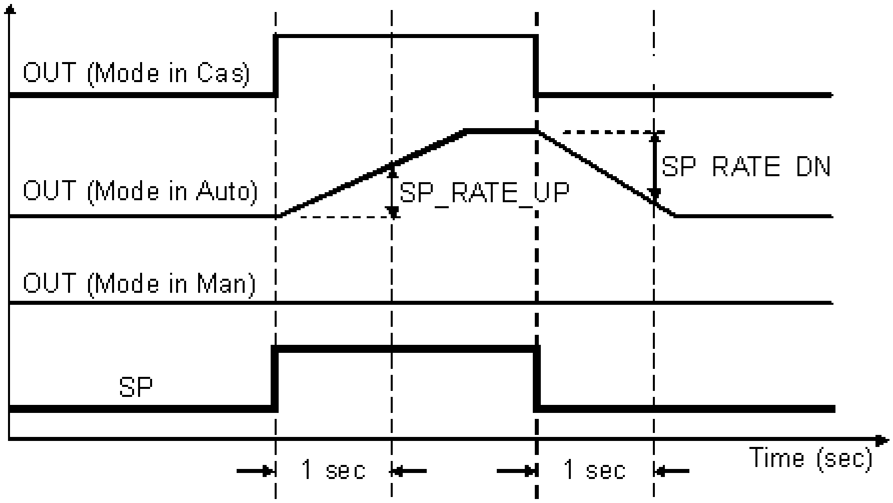
The following diagram shows the internal components of the Analog Output function block.



Analog Output Function Block Schematic Diagram

Block Execution - Analog Output Function Block

The following diagram shows the timed response of the Analog Output function block.



Analog Output Function Block Timing Diagram

You select the manner of processing the SP and the channel output value by configuring setpoint limiting options, tracking options, and conversion and status calculations.

Setpoint Limit Scaling

On download, the limits are set to the high and low end of the scale. They might be changed by subsequent overrides, which are part of the download.

On run time, the following restrictions apply:

- SP_HI_LIM is restricted to PV_SCALE_HI+.1*(PV_SCALE_HI-PV_SCALE_LO).
- SP_LO_LIM is restricted to PV_SCALE_LO-.1*(PV_SCALE_HI-PV_SCALE_LO).

If the new scale causes a limit to be outside of these rules, the DeltaV system forces the limit within the rules.

The SP parameters are not changed as a result of changing the scale or limits. However, the algorithm might change SP on the next pass.

Setpoint Selection and Limiting

You use the MODE parameter to select the source of the SP value. In Automatic (Auto) mode, the local, manually entered SP is used. In Cascade (Cas) mode, the SP comes from another block through the CAS_IN input connector. The range and units of the SP are defined by the PV_SCALE parameter.

In Manual (Man) mode, the SP automatically tracks the PV value when you select the following I/O option (IO_OPTS):

SP-PV Track in Man

This option is enabled (True) as a default. SP becomes a copy of PV when the block's target mode is Man. A manually entered SP value is overwritten on the block's next execution cycle. You can disable this option in Man or OOS mode only.

The SP value is limited to the range defined by the setpoint high limit parameter (SP_HI_LIM) and the setpoint low limit parameter (SP_LO_LIM).

In Auto mode, the rate at which a change in the SP is passed to OUT is limited by the values of the setpoint upward rate limit parameter (SP_RATE_UP) and the setpoint downward rate limit parameter (SP_RATE_DN). A limit of zero prevents rate limiting, even in Auto mode.

Conversion and Status Calculation

The working setpoint (SP_WRK) is the setpoint value after limiting. You can choose to reverse the range for conversion for fail-open actuators by selecting the following I/O option:

Increase to Close

The converted percent value is inverted and stored in the OUT parameter, except in Man mode. In Man mode, the OUT parameter is set manually.

The OUT value is used to set the analog output or the pulse discrete output of the I/O channel defined by the IO_OUT parameter. When you use a pulse discrete output I/O channel, we recommend you set the PULSE_PERIOD of the associated I/O properties to agree with the module's execution period.

You access the actuator position analog input that is associated with the output channel through the IO_READBACK parameter. The channel input is shown in the READBACK parameter (in percent) and in the PV parameter (in engineering units). When no input channel is defined, the PV and READBACK values are based on the OUT parameter.

The working setpoint (SP_WRK) is the value normally used for the BKCAL_OUT parameter. However, for those cases where the READBACK signal directly (linearly) reflects the OUT channel, you can choose to allow the PV to be used for BKCAL_OUT by selecting the following I/O option:

Use PV for BKCAL_OUT

Note You can set I/O options in Manual or Out of Service mode only.

Scaling for HART and Fieldbus Devices

XD_SCALE converts the displayed value of OUT to engineering units when the channel or port for IO_OUT is HART Analog output or Fieldbus.

Note For DVC4000/5000 series Fisher valves, you cannot write a value to XD_SCALE while the device is active. This causes a mismatch between the device and the configuration. You must take the device out of service using AMS before it can accept an XD_SCALE change.

Simulation

When simulation is enabled, the last value of OUT is maintained and is reflected in the field value of the SIMULATE parameter. In this case, the PV and READBACK values and statuses are based on the SIMULATE value and status you enter.

Initial Value and Failure Action

During I/O configuration, you can configure the following channel parameters:

FAIL_ACTION_MODE – Controls the behavior of the channel when the card goes into failure action condition due to lost communication with the controller. You can configure the channel to hold the value at the start of the failure action condition or to go to the failure action value (FAIL_ACTION_VAL).

FAIL_ACTION_VAL – The value the channel transitions to when the card goes into failure action condition, in percent. This value is used only when FAIL_ACTION_MODE is configured to set the value to FAIL_ACTION_VAL.

INIT_VAL – the value (in percent) that the channel goes to upon initial download before any function block action drives the output.

Refer to the I/O Configuration topic for more information on channel parameters.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Input failure/process variable has *Bad status* – The source of the block's process variable is bad. Indicates a hardware failure, a non-existent Device Signal Tag (DST), or a Bad status on the SIMULATE parameter.

Out of Service – The block is in Out of Service (OOS) mode.

Output failure – The write to the output failed or the DST is bad.

Readback failed – The I/O readback failed.

Simulate active – Simulation is enabled and the block is using a simulated value in its execution.

I/O Readback Options

When the channel type is HART Analog Output Channel, the following options are available for the IO_READBACK parameter:

HART_DV_SLOT0 – the slot 0 device variable value. This value is read digitally.

HART_DV_SLOT1 – the slot 1 device variable value. This value is read digitally.

HART_DV_SLOT2 – the slot 2 device variable value. This value is read digitally.

HART_DV_SLOT3 – the slot 3 device variable value. This value is read digitally.

HART_FIELD_VAL – the 4 to 20 mA signal of the HART device.

HART_PV – the Primary Variable of a HART transmitter, in engineering units. This value is read digitally.

HART_SV – the Secondary Variable of a HART transmitter, in engineering units. This value is read digitally.

HART_TV – the Tertiary Variable of a HART transmitter, in engineering units. This value is read digitally.

HART_FV – the Fourth Variable of a HART transmitter, in engineering units. This value is read digitally.

For more information on the DeltaV system’s implementation of HART communications, refer to the HART Devices and the DeltaV System topic.

Modes - Analog Output Function Block

The Analog Output function block supports multiple modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

Cascade (Cas)

Remote Cascade (RCas)

The target mode of the block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Analog Output Function Block

I/O card operating condition and wiring fault detection are reflected in the status of PV, OUT, and BKCAL_OUT. A limited SP condition is reflected in the BKCAL_OUT status. When simulation is enabled through the SIMULATE parameter, you can set the value and status for PV and READBACK.

When the block is in Cas mode and the CAS_IN input goes bad, the block sheds mode to the next permitted mode.

Parameters - Analog Output Function Block

The following table lists the system parameters for the Analog Output function block:

Analog Output Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BKCAL_OUT	EU of PV_SCALE	The value and status required by the BKCAL_IN input of another block to prevent reset windup and to provide bumpless transfer to closed loop control.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Analog Output block are Simulate active, Input failure/ process variable has *Bad* status, Output failure, Readback failed, and Out of Service.

Parameter	Units	Description
CAS_IN	EU of PV_SCALE	The remote SP value from another function block.
CHANNEL	None	The number of the logical hardware channel that is connected to the I/O block. It defines the transducer to be used going to or from the physical world.*
FSTATE_TIME	Seconds	The time in seconds from detection of fault of the output block remote setpoint to the output action of the block output if the condition still exists.*
FSTATE_VAL	EU of PV	The preset analog SP value to use when fault occurs. Ignored if the I/O option Fault State to value is false.*
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 only if the following conditions are true: - The Write To Inspect Alarm context menu item has been selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition exists for Variability only if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
IO_OPTS	None	I/O options. Allow you to select how the I/O signals are processed. The supported I/O options for the Analog Output function block are SP_PV Track in Man, Increase to Close, and Use PV for BKCAL_OUT.
IO_OUT	None	Defines the output DST for the block.
IO_READBACK	None	Defines the Device Signal Tag (DST) for the input channel that provides readback for the value written to the channel defined by IO_OUT.
MODE	None	Parameter used to request and show the source of the setpoint and/or output used by the block.
OUT	Percent	The primary value and status calculated by the block in Auto mode. OUT can be set manually in Man mode.
PV	EU of PV_SCALE	The process variable used in block execution and alarm limit detection.
PV_SCALE	None	The high and low scale values, the engineering units code, and the number of digits to the right of the decimal point associated with the PV.

Parameter	Units	Description
RCAS_IN	EU of SP_SCALE	The remote analog setpoint value and status. Input provided by a device or the output of another block.
RCAS_OUT	EU of PV_SCALE	The remote analog setpoint value and status after ramping. Output provided to a device for back calculation and to allow action to be taken under limiting conditions or mode change.
READBACK	Percent	The percent value of the measured or implied actuator position associated with the OUT value.
SHED_OPT	None	Defines action to be taken on remote control device time out.
SHED_TIME	Seconds	The maximum allowable time between RCAS_IN and ROUT_IN updates. If exceeded, mode shedding takes place.
SIMULATE	Percent	Enables simulation and allows you to enter an input value and status. When simulation is enabled, the block does not attempt to write to the hardware.
SP	EU of PV_SCALE	The block's setpoint value.
SP_HI_LIM	EU of PV_SCALE	The highest SP value allowed.
SP_LO_LIM	EU of PV_SCALE	The lowest SP value allowed.
SP_RATE_DN	EU of PV_SCALE per second	Ramp rate at which downward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_RATE_UP	EU of PV_SCALE per second	Ramp rate at which upward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_WRK	EU of PV_SCALE	The working setpoint of the block. It is the result of setpoint rate-of-change limiting. The value is converted to percent to obtain the block's OUT value.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*

Parameter	Units	Description
STDEV_CAP	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.
STDEV	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The standard deviation of PV. For analog control blocks in AUTO, mean is assumed to be the SP. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The status options are Target to Manual if Bad IN and BAD if Limited. Each function block uses one or none of the status options.*
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.*
XD_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the channel input value.

* These parameters are only visible when the function block is extended to a Fieldbus device.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Analog Output Function Block

The way you configure an Analog Output function block and its associated input and output channels and parameters depends on whether it has been assigned to a fieldbus device and whether the field device requires a traditional 4 to 20 mA input, a 4 to 20 mA input using HART Communications Protocol, or a discrete pulse duration input.

AO Block Assigned to a Fieldbus Valve

If a fieldbus valve loses air, or a fieldbus valve positioner loses power, the valve will either go fully open or fully closed. This state is determined by the valve itself and is often termed its shelf position or air fail position. This state is different from the Fault State parameter (FSTATE_VAL) of the Analog Output function block when it is assigned to a fieldbus valve.

By default the Analog Output function block's FSTATE_VAL and FSTATE_TIME parameters are set to 0. This causes the block to hold the last value when communication on the fieldbus is lost. It is highly recommended that you configure the FSTATE_VAL parameter to correctly set the valve position if fieldbus communications to the valve cease for the time period that you configure with the FSTATE_TIME parameter.

Note The FSTATE_VAL and FSTATE_TIME parameters only appear in the parameter list after the block has been assigned to a fieldbus device.

Limit Write Requests to Static or Non-Volatile Parameters

It is recommended that you limit the number of periodic writes to all static or non-volatile parameters, such as HI_HI_LIM, LOW_CUT, SP, TRACK_IN_D, OUT, IO_OPTS, BIAS, STATUS_OPTS, SP_HI_LIM, and so on. Static parameter writes increment the static revision counter, ST_REV, and are written to the device's non-volatile memory. Fieldbus devices have a non-volatile memory write limit. If a static or non-volatile parameter is configured to be written periodically, the device can stop its normal operation after it reaches its limit or fail to accept new values. Consult the device documentation to determine if a parameter is static or non-volatile.

Another way to configure your system to avoid burning out non-volatile memory is to link a CALC block or PID block running in the controller to the AO block running in the device. This method avoids the use of asynchronous writes by writing to the AO block through a publisher/subscriber mechanism.

To link a CALC block to the AO block:

1. Set the AO block mode to CAS, AUTO, or MAN.
2. Link the CALC block's OUT parameter to the AO block's CAS_IN parameter.

To link a PID block to the AO block:

1. Set the AO block mode to CAS, AUTO, or MAN.
2. Link the PID block's OUT parameter to the AO block's CAS_IN parameter.
3. Link the AO block's BKCAL_OUT parameter to the PID block's BKCAL_IN parameter.
4. Set the PID parameter CONTROL_OPTS to Track Enable.
5. Write a value to the AO block using the TRK_IN_D and TRK_VAL parameters.

Actuator Requiring Traditional 4 to 20 mA Input

When an actuator is designed to accept a 4 to 20 mA signal, use the AO Card, 4 to 20 mA for the output channel. You configure the associated output channel and the Analog Output block as follows:

Channel Type: Analog Output Channel

Analog Output block IO_OUT parameter: Select OUT.

Analog Output block IO_READBACK parameter: Select FIELD_VAL_PCT when the input is 4 to 20 mA.

Analog Output block PV_SCALE parameter: Set the range and engineering units to values that correspond to the Analog Output block's 4 to 20 mA output. In many cases, PV_SCALE is set to 0 – 100%.

Analog Output block IO_OPTS parameter: Select Increase to Close when the actuator is designed to fail open.

If you are using the CAS_IN connector wired from another block, the Analog Output block's BKCAL_OUT parameter should be wired to the other block's BKCAL_IN parameter.

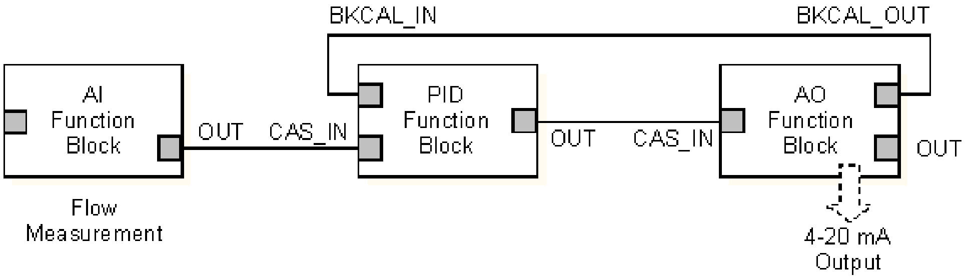
Application Example: **Actuator Requiring Traditional 4 to 20 mA Input**

Assume a regulating valve equipped with an air-operated actuator interfaces to the analog output channel through an I/P transducer that requires 4 to 20 mA input. The Analog Output (AO) function block is used with an Analog Input (AI) function block and a PID function block to control flow in a pipe. The configuration differs depending on whether the valve actuator with the I/P transducer is designed to allow the valve to fail closed or to fail open on the loss of the 4 to 20 mA signal. The following table shows configuration settings for each of these cases.

Analog Output Function Block Configuration Example for an Actuator Requiring 4 to 20 mA Input

Configuration Setting	Fail Close	Fail Open
Channel Type	Analog Output	Analog Output
IO_OUT	OUT	OUT
PV_SCALE	0 to 100%	0 to 100%
IO_OPTS Increase to Close	Not selected	Selected

The following diagram is the function block diagram for this example.



Analog Output Function Block Diagram Example for an Actuator Requiring 4 to 20 mA Input

Actuator Requiring HART Communications Protocol 4 to 20 mA Input

When an actuator is designed to accept a 4 to 20 mA signal using the HART Communications Protocol, use the AO Card, 4 to 20 mA, HART for the output channel. Configure the associated output channel and the Analog Output block as follows:

Channel Type: HART Analog Output Channel

Analog Output block IO_OUT parameter: Select OUT.

Analog Output block IO_READBACK parameter: Select the HART device or dynamic variable that holds the desired value. Define the available variables through the DeltaV Explorer.

Analog Output block PV_SCALE parameter: Set the range and engineering units to values that correspond to the Analog Output block's 4 to 20 mA output. In many cases, PV_SCALE is set to 0 – 100%.

Analog Output block XD_SCALE parameter: Set the range and engineering units to values that correspond to PV_SCALE. Set the engineering units to match those appropriate to the field device.

Analog Output block IO_OPTS parameter: Select Increase to Close when the actuator is designed to fail open.

If you are using the CAS_IN connector wired from another block, the Analog Output block's BKCAL_OUT parameter should be wired to the other block's BKCAL_IN parameter.

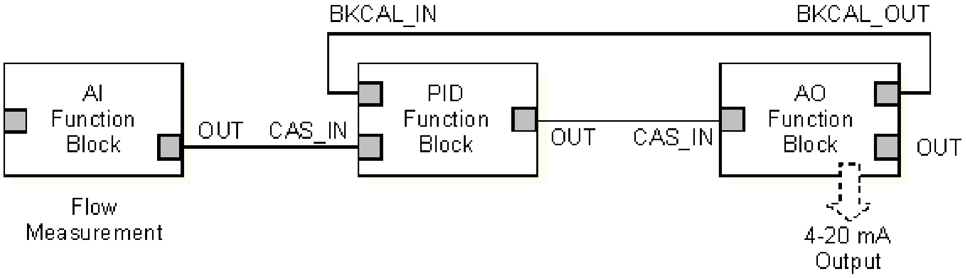
Application Example: **Actuator Requiring 4 to 20 mA Input Using HART Communications Protocol**

Assume a regulating valve and actuator that communicates using the HART Communications Protocol interfaces to the analog output channel. The Analog Output (AO) function block is used with an Analog Input (AI) function block and a PID function block to control flow in a pipe. The configuration differs depending on whether the valve actuator with the I/P transducer is designed to allow the valve to fail closed or to fail open on the loss of the 4 to 20 mA signal. The following table shows configuration settings for each of these cases.

Analog Output Function Block Configuration Example for an Actuator Requiring 4 to 20 mA Input

Configuration Setting	Fail Close	Fail Open
Channel Type	HART Analog Output	HART Analog Output
IO_OUT	OUT	OUT
PV_SCALE	0 to 100%	0 to 100%
IO_OPTS Increase to Close	Not selected	Selected

The following diagram is the function block diagram for this example:



Analog Output Function Block Diagram Example for an Actuator Requiring 4 to 20 mA Input

Field Device Requiring Discrete Pulse Duration Input

When a field device, such as a heater band, is designed to accept a pulse duration input, you use a DO card type for the output channel and you configure the output channel and Analog Output function block as follows:

Channel Type: Continuous Pulse Output Channel. Define the PULSE_PERIOD (in seconds) equal to the AO block execution period. Note that PULSE_PERIOD will probably be different than the module scan rate.

Analog Output block IO_OUT parameter: Select ON_TIME.

Analog Output block IO_READBACK parameter: Select FIELD_VAL_PCT when the input is 4 to 20mA. (Refer to the Analog Input function block information for HART input options.)

Analog Output block PV_SCALE parameter: Set the range and engineering units to the values that correspond to the Analog Output block's output. In many cases, PV_SCALE is set to 0 – 100%.

Analog Output block IO_OPTS parameter: Do not select Increase to Close.

When the Analog Output function block is used for pulse duration output through a discrete channel, the OUT value (in percent) is used to determine the time on during the defined pulse period. For example, when OUT = 50%, the discrete output channel is turned on for 50% of the defined PULSE_PERIOD. The discrete output is turned on by the discrete card and is accurate to within 2 msec of the value determined by OUT and PULSE_PERIOD.

Application Example: **Field Device Requiring Pulse Duration Input**

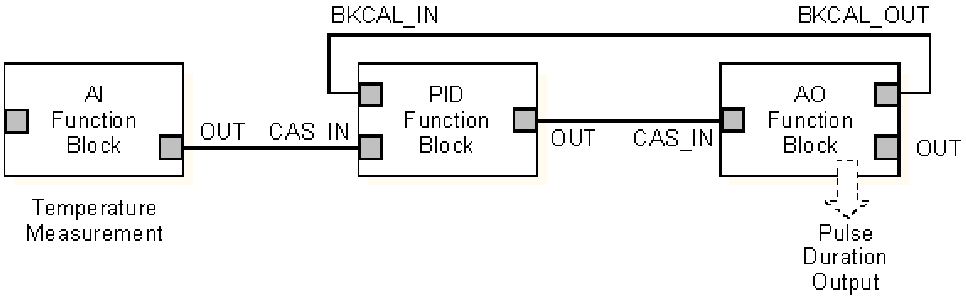
Assume a heater band on an extruder interfaces to the discrete output channel through a contactor that requires a pulse duration discrete input. The Analog Output (AO) function block is used with an Analog Input (AI) function block and a PID function block to control temperature by regulating duty cycle (percent time on).

The module containing these blocks is defined to execute on a ten-second period. With continuous heat applied (100% duty cycle), the energy input is 50 kW. The configuration differs depending on whether the AO block setpoint is to be set in kW or in percent energy input. The following table shows configuration settings for each of these cases.

Analog Output Function Block Configuration Example for an Actuator Requiring Pulse Duration Input

Configuration Setting	Setpoint in Percent	Setpoint in kW
Channel Type	Continuous Pulse Output, 10 sec	Continuous Pulse Output, 10 sec
AO Block IO_OUT Parameter	ON_TIME	ON_TIME
PV_SCALE	0 to 100%	0 to 50 kW
IO_OPT Increase to Close	Not selected	Not selected

The following figure is the function block diagram for this example:



Analog Output Function Block Diagram for Actuator Requiring Pulse Duration Input

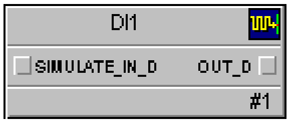
Discrete Input (DI) Function Block

The Discrete Input (DI) function block accesses a single discrete input from a two-state field device and makes the processed physical input available to other function blocks. You can configure inversion and alarm detection on the input value.

The Discrete Input function block supports block alarming, mode control, signal status propagation, and simulation.

Normally, the block is used in Automatic (Auto) mode so that the process variable (PV_D) is copied to the output (OUT_D). You can change the mode to Manual (Man) to disconnect the field signal and substitute a manually entered value for OUT_D. In this case, PV_D continues to show the value that will become OUT_D when the mode is changed to Auto.

To support testing, you can enable simulation. This allows the measurement value to be supplied manually or from another block through the SIMULATE_IN_D input.



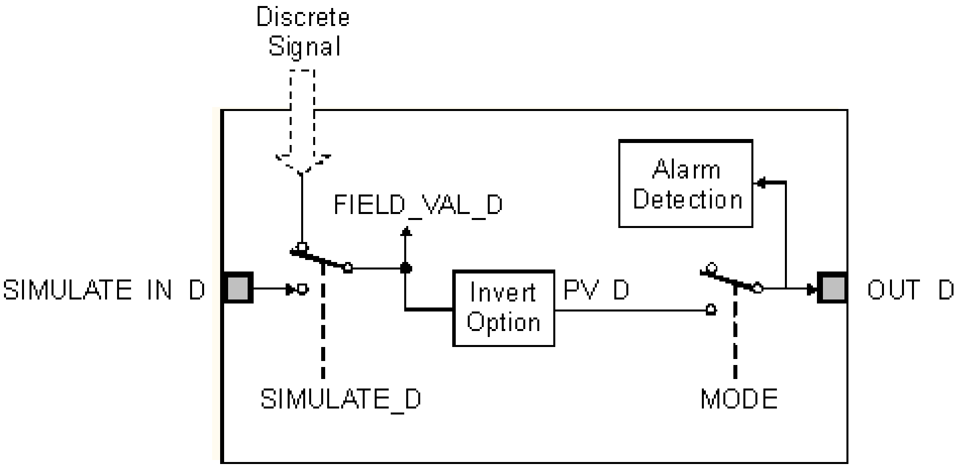
DI Function Block with Simulation Enabled

SIMULATE_IN_D is the simulated discrete value used by the block when simulation is enabled.

OUT_D is the discrete output value and status.

Schematic Diagram - Discrete Input Function Block

The following diagram shows the internal components of the Discrete Input function block:



Discrete Input Function Block Schematic Diagram

Block Execution - Discrete Input Function Block

I/O Selection

When you configure the Discrete Input function block, you select the I/O channel associated with the discrete measurement by configuring the Device Signal Tag (DST) of the IO_IN parameter. You select the Device Tag and the parameter the Discrete Input block accesses on that channel.

When you select Discrete Input Channel for the channel type, the only selectable channel parameter is:

FIELD_VAL_D – The last Boolean value reported by the card and the current status of the channel.

You configure contact debounce filtering during I/O configuration.

Simulation

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block.

During configuration, decide whether you want to enter the simulated value/status manually during operation or use a value/status from another block for the simulated value/status.

When the value is entered manually:

- The operator first enables simulation by selecting the SIMULATE_D parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.
- If SIMULATE_IN_D is not connected (status = *Bad: NotConnected*), the operator enters the value to be used in the SIMULATE_D parameter Simulate Value field. In online operation, the operator can enter a simulated status value in the Simulate Status field.

Note Make sure SIMULATE_IN_D is not connected if you want to enter the value or status manually. When SIMULATE_IN_D is connected, the value from the SIMULATE_IN_D Value field is used as the simulated value.

When the value/status from another block is used:

- During configuration, connect SIMULATE_IN_D to the desired block output or parameter. Do not enter a value in the Simulate Value field of the SIMULATE_IN_D input; the block uses the connected value automatically.
- During operation, the operator enables simulation by selecting the SIMULATE_D parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.

Note Do not enter a value for the SIMULATE_IN_D parameter. If you do and the status of SIMULATE_IN_D is not *Bad: NotConnected*, the manually entered value for SIMULATE_IN_D overrides any value you enter in SIMULATE_D.

When SIMULATE_D is **not** enabled, the hardware value becomes the FIELD_VAL_D parameter value in the block. When SIMULATE_D **is** enabled, the value and status of the SIMULATE_IN_D parameter become the value and status of FIELD_VAL_D.

Field Value Processing

You can configure the Invert I/O option (IO_OPTS) to process FIELD_VAL_D:

Invert

The output of the Invert processor is PV_D. This value goes to the mode switch where it becomes OUT_D when the mode is Auto. OUT_D is tested for an alarm state. You might choose this option when the field contact is normally closed, so an open contact or a broken wire represents the active state of the condition being sensed.

PV Filter

Filtering in a Discrete Input function block is performed with on and off delays. FIELD_VAL_D must be on (or off) for PV_FTIME without changing before the block passes a change to PV_D.

Note You can set the I/O option in Manual or Out of Service mode only.

Alarm Detection - Discrete Input Function Block

You configure the DISC_LIM parameter to select the state of the output (OUT_D) that causes the alarm condition parameter (DISC_ACT) to be True and to set Discrete Alarm substatus in the output (OUT_D). You can enter any value between 0 and 255, although only 0 and 1 produce an alarm. However, you can enter state 255 to indicate that you never want an alarm indication.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Simulate active – SIMULATE_D is enabled; therefore, OUT_D is not real.

Input failure/process variable has *Bad status* – The source of the block's process variable is bad.

Out of Service – The block is not being processed.

Modes - Discrete Input Function Block

The Discrete Input function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Discrete Input Function Block

Under normal conditions, a *Good: Non-cascade* status is passed through to OUT_D. However, the block supports a status action on failure and block error indications.

For more information regarding status options, refer to the Status Options topic.

Action on Failure

When the hardware goes bad, FIELD_VAL_D, PV_D, and OUT_D are set to *Bad* status and the BLOCK_ERR parameter shows Bad PV. When SIMULATE_D is enabled, FIELD_VAL_D, PV_D, and OUT_D are set to the simulation status. When the block is set to Man mode, OUT_D is set to *Good: Non-cascade, Constant* status.

Parameters - Discrete Input Function Block

The following table lists the system parameters for the Discrete Input function block:

Discrete Input Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).

Parameter	Units	Description
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Discrete Input function block are Simulate active, Input failure/process variable has *Bad* status, and Out of Service.
CHANNEL	None	The number of the logical hardware channel that is connected to the I/O block. It defines the transducer to be used going to or from the physical world.*
DISC_ACT	None	The result of alarm detection associated with DISC_LIM. If the state of OUT_D matches the value of DISC_LIM, DISC_ACT equals True.
DISC_LIM	None	The state of the discrete input that causes an alarm. Any number from 0 to 255 can be entered, although only 0 and 1 produce an alarm. State 255 specifies that no alarm indication is to be shown.
FIELD_VAL_D	None	The value and status of the discrete input from a field device.
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 only if the following conditions are true: - The Write to Inspect Alarm context menu item has been selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition exists for Variability only if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
IO_IN	None	Defines the input DST for the I/O channel used for the PV.
IO_OPTS	None	I/O options. Allow you to select options for I/O value processing. The supported I/O option for the Discrete Input function block is Invert.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.
OUT_D	None	The discrete output value and status.
OUT_STATE	None	Index to the text describing the states of a discrete for the value obtained from the transducer.*

Parameter	Units	Description
PV_D	None	The discrete process variable used in block execution.
PV_FTIME	Seconds	The time that FIELD_VAL_D must be on or off before PV_D changes.
SIMULATE_D	None	Enables simulation and allows you to enter an input value and status when SIMULATE_IN_D is not connected.
SIMULATE_IN_D	None	The simulated discrete input value used when simulation is enabled.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The status options are Target to Manual if Bad IN and BAD if Limited. Each function block uses one or none of the status options.
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.*
SUBSTITUTE_IN_D	None	The substituted (from a field device) discrete input value and status.*
XD_STATE	None	Index to the text describing the states of a discrete for the value obtained from the transducer.*

* These parameters are only visible when the function block is extended to a Fieldbus device.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

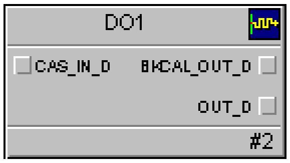
Discrete Output (DO) Function Block

The Discrete Output (DO) function block takes a binary setpoint and writes it to a specified I/O channel to produce an output signal. You can confirm the physical output operation by configuring a hardware discrete input, which produces a value that should match the setpoint.

The Discrete Output function block supports mode control, output tracking, and simulation. There are no standard alarms in this function block. Custom alarms are supported.

Normally, the block is used in Cascade (Cas) mode so a signal from another block changes the setpoint. You change to Automatic (Auto) mode to disconnect the other block and to set the setpoint locally. The other block tests the status of the Discrete Output function block through the BKCAL_OUT_D output.

To support testing, the block uses a simulation switch to indicate *Good* block status even when there is no hardware connected.



DO Function Block Used in a Simulation

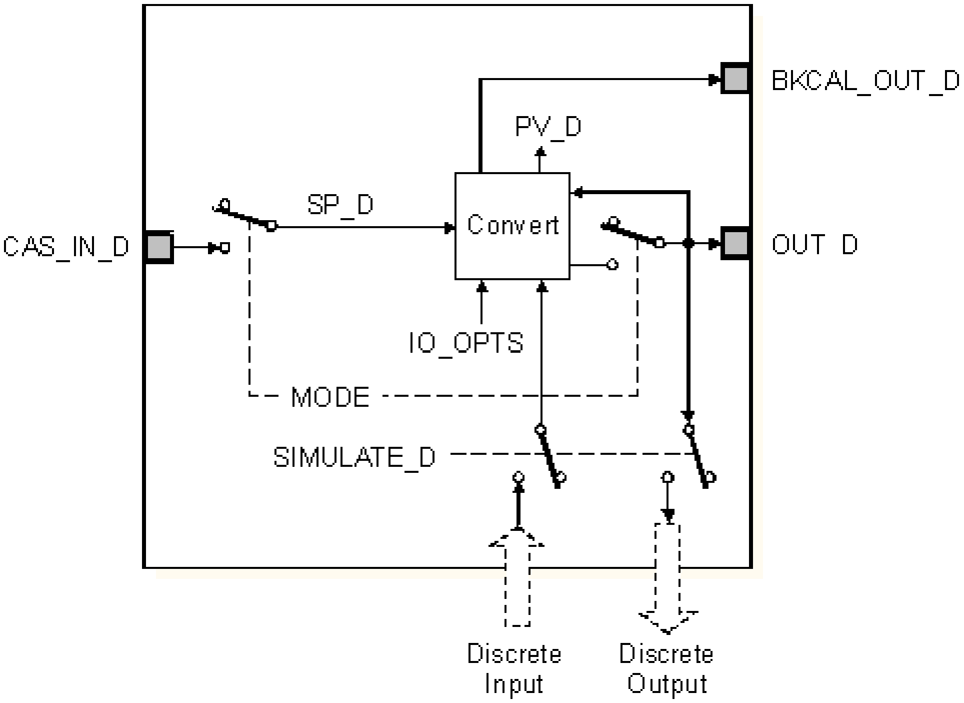
CAS_IN_D is the remote setpoint value from another function block.

BKCAL_OUT_D is the output value and status required by the BKCAL_IN_D input of another block for output tracking.

OUT_D is the discrete output value and status.

Schematic Diagram - Discrete Output Function Block

The following diagram shows the internal components of the Discrete Output function block:



Discrete Output Function Block Schematic Diagram

Block Execution - Discrete Output Function Block

The Discrete Output function block determines its setpoint, sets the output, and, optionally, checks a feedback signal from the hardware to confirm the physical output operation.

Setting the Output

The block first must determine its setpoint. Normally, the block is used in Cas mode and the cascade input value (CAS_IN_D) becomes the setpoint (SP_D). In Automatic (Auto) or Manual (Man) mode, the block disconnects CAS_IN_D and allows a manual entry to change the value of SP_D.

Note CAS_IN_D should always be a discrete value. When CAS_IN_D is an analog value, block behavior is unpredictable.

You can select setpoint tracking by configuring the SP_PV Track in Man I/O option (IO_OPTS).

SP-PV Track in Man

This option is enabled (True) as a default. SP_D becomes a copy of PV_D when the block's target mode is Man. A manually entered SP_D value is overwritten on the block's next execution cycle. This can prevent a state change when transitioning from Man to Auto mode. You can disable this option in Man or OOS mode only.

You can select setpoint inversion by configuring the Invert I/O option:

Invert

When this option is enabled (True), OUT_D becomes an inverted copy of SP_D. When this option is not enabled (False), OUT_D is a straight copy of SP_D.

When you configure the Discrete Output function block, you select the I/O channel associated with a discrete output by configuring the Device Signal Tag (DST) of the IO_OUT parameter. You select the Device Tag and the parameter the DO block accesses on that channel.

If you select Discrete Output Channel for the channel type, the only selectable channel parameter is:

OUT_D – The current Boolean value driven to the card and the status of the output channel.

If you select Momentary Output Channel for the channel type, the only selectable channel parameter is:

OUT_D – Writing a True (1) value to this parameter causes the channel to output a pulse of a fixed duration. You configure the pulse duration during I/O configuration.

You configure a failure action and value for the output during I/O configuration.

Simulation

When SIMULATE_D is not enabled, and the mode is not Out of Service (OOS), the value of OUT_D is sent to the hardware. When SIMULATE_D is enabled, the status of OUT_D is forced to *Good* regardless of the condition of the hardware.

Getting Feedback from the Hardware

You can configure the IO_READBACK parameter to determine the pair of field wires that are used for confirming the hardware output of this block. To select the DST, you enter the Device Tag of the desired discrete input, or you can browse for it among all of the available Device Tags. There is only one possible parameter of the Device Tag that can be used (FIELD_VAL_D), so it is entered automatically.

When IO_READBACK is not configured, a copy of OUT_D is used in its place (with a delay of one execution time). This signal goes through the SIMULATE_D switch to become READBACK_D. You can enter a value and status for READBACK_D in SIMULATE_D when it is enabled. The readback value is processed through the Invert I/O option to become PV_D, which normally matches SP_D in Auto or Cas mode.

Usually, BKCAL_OUT_D has the value of SP_D and the status of the hardware. When SIMULATE_D is enabled, the status is forced to *Good: Cascade*. You can configure the Use PV for BKCAL_OUT I/O option:

Use PV for BKCAL_OUT

Note You can set I/O options in Manual or Out of Service mode only.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Simulate active – SIMULATE_D is enabled; therefore, PV_D is not real.

Input failure/process variable has *Bad status* – The source of the block's process variable is bad. For the DO block, this might indicate a hardware failure, a non-existent Device Signal Tag (DST), or a Bad status on the SIMULATE_D parameter.

Output failure – The output hardware or the DST is bad.

Readback failed – The hardware or DST providing readback is bad.

Out of Service – The block is not being processed.

Modes - Discrete Output Function Block

The Discrete Output function block supports five modes:

Initialization Manual (IMan)

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

Cascade (Cas)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Discrete Output Function Block

Under normal conditions, the output statuses (OUT_D and BKCAL_OUT_D) are *Good: Cascade*. However, the block supports a status action on failure and block error indications.

Action on Failure

When the output hardware goes bad, the OUT_D and BKCAL_OUT_D statuses are set to *Bad: DeviceFail*. The BLOCK_ERR parameter shows Output failure. When SIMULATE_D is enabled, OUT_D does not show *Bad* status.

When there is no DST assigned to the IO_READBACK parameter, PV_D status is set to *Bad* and the BLOCK_ERR parameter shows Bad PV.

When the input hardware used for IO_READBACK goes bad, BLOCK_ERR shows Bad PV and Readback failed. In this case, the status of READBACK_D and PV_D is *Bad: DeviceFail*.

Parameters - Discrete Output Function Block

The following table lists the system parameters for the Discrete Output function block:

Discrete Output Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BKCAL_OUT_D	None	The value and status required by the BKCAL_IN_D input of another block for output tracking.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Discrete Output function block are Simulate active, Input failure/process variable has *Bad* status, Output failure, Readback failed, and Out of Service.
CAS_IN_D	None	The remote setpoint value from another block.
CHANNEL	None	The number of the logical hardware channel that is connected to the I/O block. It defines the transducer to be used going to or from the physical world.*
FSTATE_TIME	Seconds	The time in seconds from detection of fault of the output block remote setpoint to the output action of the block output if the condition still exists.*
FSTATE_VAL_D	EU of PV	The preset discrete SP_D value to use when fault occurs. Ignored if the I/O option Fault State to value is false.*

Parameter	Units	Description
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 only if the following conditions are true: - The Write To Inspect Alarm context menu item has been selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition exists for Variability only if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
IO_OPTS	None	I/O options. Allow you to select how the I/O signals are processed. The supported I/O options for the Discrete Output function block are SP_PV Track in Man, Invert, and Use PV for BKCAL_OUT.
IO_OUT	None	Defines the output DST for the block.
IO_READBACK	None	Defines the Device Signal Tag (DST) for the input channel that provides readback for the value written to the channel defined by IO_OUT.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.
OUT_D	None	The discrete output value and status.
PV_D	None	The discrete process variable calculated from READBACK_D.
PV_STATE	PV	The readback indication of the current output.*
RCAS_IN_D	EU of SP_SCALE	The remote discrete setpoint value and status. Input provided by a device or the output of another block.*
RCAS_OUT_D	EU of PV_SCALE	The remote discrete setpoint value and status after ramping. Output provided to a device for back calculation and to allow action to be taken under limiting conditions or mode change.*
READBACK_D	None	The discrete process value from IO_READBACK.
SHED_OPT	None	Defines action to be taken on remote control device time out.*
SIMULATE_D	None	Enables simulation.
SP_D	None	The discrete target block output value (setpoint).
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*

Parameter	Units	Description
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The status options are Target to Manual if Bad IN and BAD if Limited. Each function block uses one or none of the status options.*
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.*
XD_STATE	None	Index to the text describing the states of a discrete for the value obtained from the transducer.*

* These parameters are only visible when the function block is extended to a Fieldbus device.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Multiple Discrete Input (FFMDI) Function Block

The Multiple Discrete Input (FFMDI) function block combines the eight channels of a Discrete Input card and makes them available as an 8-bit input to other function blocks.

This block must be assigned to an H1 Carrier device. The carrier supports one Discrete Output and one Discrete Input card. The Multiple Discrete Input block processes the discrete input values of multiple devices through a Discrete Input card connected to the carrier. Using the Multiple Discrete Input block assigned to the carrier enables multiple discrete devices to communicate back to the DeltaV controller through the H1 fieldbus rather than through conventional wiring.

The Multiple Discrete Input function block supports mode control, signal status propagation, and simulation.

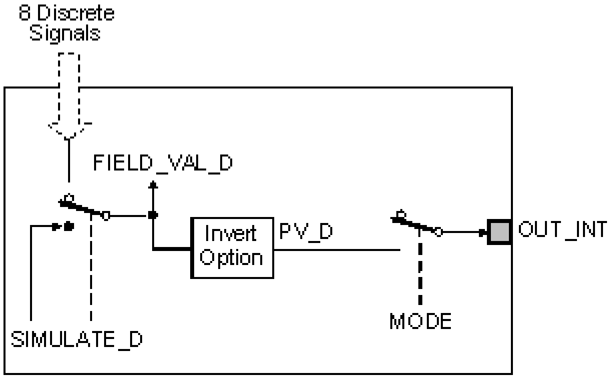
Normally, the block is used in Automatic (Auto) mode so that the process variable (PV_D) is copied to the output (OUT_INT). You can change the mode to Manual (Man) and substitute a manually entered value for OUT_INT. In this case, PV_D continues to show the value that will become OUT_INT when the mode is changed to Auto.

Multiple Discrete Input Function Block

OUT_INT is the 8-bit integer output value and status.

Schematic Diagram - Multiple Discrete Input Function Block

The following diagram shows the internal components of the Multiple Discrete Input function block:



Multiple Discrete Input Function Block Schematic Diagram

Block Execution - Multiple Discrete Input Function Block

I/O Selection

When you configure the Multiple Discrete Input function block, you assign the block to an FFMDI block in an H1 Carrier device. The carrier device, when fully installed, has a Discrete Input card and a Discrete Output card connected to it. Assigning the block to the carrier connects FIELD_VAL_D to the eight channels on the Discrete

Input card. FIELD_VAL_D is the last 8-bit integer value reported by the card and the current status of the carrier. Refer to the Status Handling – Multiple Discrete Input Function Block topic for more details.

Simulation

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually.

When the value is entered manually:

- The operator first enables simulation by selecting the SIMULATE_D parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.
- The operator enters the value to be used in the SIMULATE_D parameter Simulate Value field. In online operation, the operator can enter a simulated status value in the Simulate Status field.

When SIMULATE_D is not enabled, the hardware value becomes the FIELD_VAL_D parameter value in the block. When SIMULATE_D is enabled, the simulated value and status becomes the value and status of FIELD_VAL_D.

Field Value Processing

Invert – You can configure the Invert I/O option (IO_OPTS) to process FIELD_VAL_D.

Selecting the invert option causes all channel values to be inverted. The output of the Invert processor is PV_D. You might choose this option when the field contact is normally closed so that an open contact or a broken wire represents the active state of the condition being sensed.

Note You can set the I/O option in Out of Service mode only.

PV Filter – Filtering in a Multiple Discrete Input function block is performed with on and off delays. FIELD_VAL_D must be on (or off) for PV_FTIME without changing before the block passes a change to PV_D. PV_FTIME is applicable for all of the channels.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Simulate active – SIMULATE_D is enabled; therefore, OUT_INT is not real.

Input failure/process variable has *Bad status* – The source of the block's process variable is bad.

Out of Service – The block is not being processed.

Other Error – The block is not communicating with the carrier.

Modes - Multiple Discrete Input Function Block

The Multiple Discrete Input function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Multiple Discrete Input Function Block

Under normal conditions, a *Good: Non-cascade* status is passed through to OUT_INT.

FIELD_VAL_D, PV_D, and OUT_INT are set to *Bad* status, and the BLOCK_ERR parameter shows Bad PV when one of the following conditions is met:

- The card is not present.
- The incorrect card type is present.
- The transducer block or the resource block is Out of Service.
- Communication to the DI card is bad.

When SIMULATE_D is enabled, FIELD_VAL_D, PV_D, and OUT_INT are set to the simulation status. When the block is set to Man mode, OUT_INT is set to *Good: Non-cascade, Constant* status, unless the status option (STATUS_OPTS) of Uncertain if Man Mode is set.

For more information regarding status options, refer to the Status Options topic.

Parameters - Multiple Discrete Input Function Block

The following table lists the system parameters for the Multiple Discrete Input function block.

Multiple Discrete Input Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Multiple Discrete Input function block are Simulate active, Input failure/process variable has *Bad* status, Out of Service, and Other Error.
FIELD_VAL_D	None	The 8-bit integer value and status of the discrete input from the H1 carrier.

Parameter	Units	Description
IO_OPTS	None	I/O options. Allow you to select options for I/O value processing. The supported I/O option for the Multiple Discrete Input function block is Invert.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.
OUT_INT	None	The 8-bit integer output value and status.
PV_D	None	The 8-bit integer process variable used in block execution.
PV_FTIME	Seconds	The time that each bit of FIELD_VAL_D must be on or off before the corresponding bit of PV_D changes.
SIMULATE_D	None	Enables simulation and allows you to enter a discrete value and status.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written ,but the value is not changed.
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The applicable status option for this block is Uncertain if Man mode. When this option is selected and the fault state is Active, the substatus on BKCAL_OUT is Fault State Active.
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.
SUBSTITUTE_IN_D	None	The substituted (from a field device) discrete input value and status. This parameter is used to provide user-entered value and status to OUT_INT when either the device placeholder is not commissioned or the device is not communicating with the H1 card.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Multiple Discrete Input Block

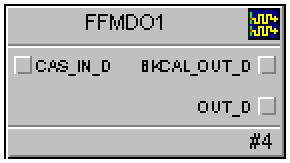
The application information for the Multiple Discrete Output function block describes the use of both the Multiple Discrete Output and the Multiple Discrete Input function blocks.

Multiple Discrete Output (FFMDO) Function Block

The Multiple Discrete Output (FFMDO) function block takes an 8-bit setpoint and writes it to the I/O channels of a Discrete Output card on an H1 Carrier device. Each of the eight bits represents a binary setpoint for a single channel in the Discrete Output card. This block must be assigned to an H1 Carrier Device. The block is used to produce discrete output values for multiple devices through a Discrete Output card connected to an H1 carrier. Using the Multiple Discrete Output block assigned to a carrier enables the DeltaV controller to communicate with multiple discrete devices through the H1 fieldbus rather than through conventional wiring.

The Multiple Discrete Output function block supports mode control and simulation. There are no standard alarms in this function block.

Normally, the block is used in Cascade (Cas) mode so that a signal from another block changes the setpoint. Change to Automatic (Auto) mode to disconnect the other block and to set the setpoint locally.



Multiple Discrete Output Function Block

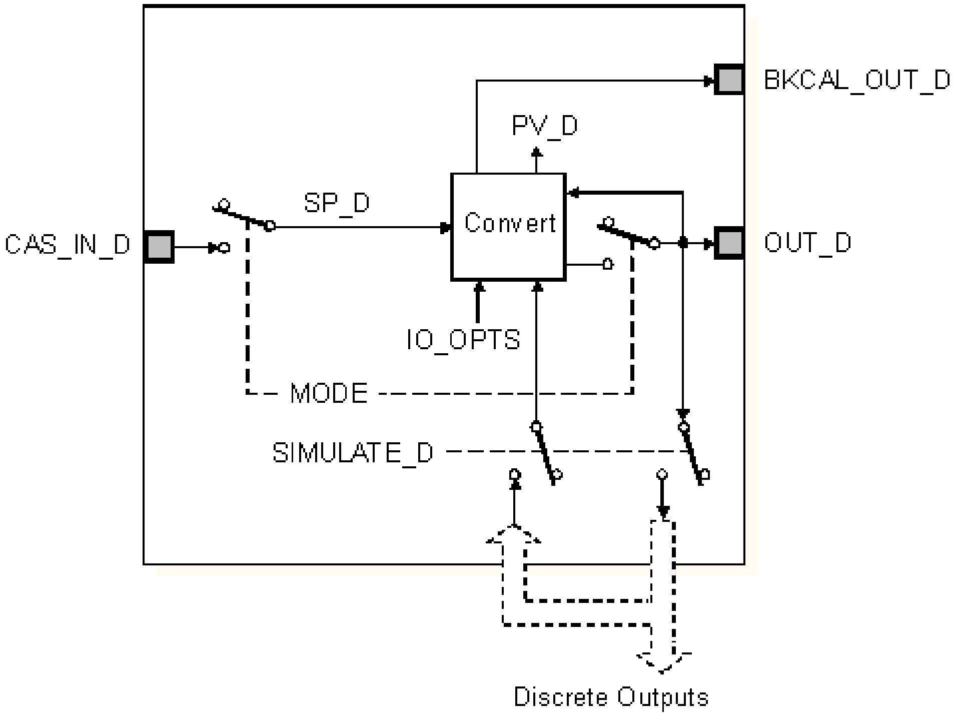
CAS_IN_D is the remote setpoint value from another function block.

BKCAL_OUT_D is the output value and status required by the BKCAL_IN_D input of another block.

OUT_D is the 8-bit integer output value and status.

Schematic Diagram - Multiple Discrete Output Function Block

The following diagram shows the internal components of the Multiple Discrete Output function block.



Multiple Discrete Output Function Block Schematic Diagram

Block Execution - Multiple Discrete Output Function Block

The Multiple Discrete Output function block determines its setpoint and sets the output.

Setting the Output

The block first must determine its setpoint. Normally, the block is used in Cas mode, and the cascade input value (CAS_IN_D) becomes the setpoint (SP_D). In Automatic (Auto) mode, the block disconnects CAS_IN_D and allows a manual entry to change the value of SP_D. In Man mode, OUT_D can be entered manually.

Note CAS_IN_D should always be an 8-bit integer value.

I/O Options

You select the I/O options by using the IO_OPTS parameter:

SP-PV Track in Man – This option is enabled (True) as a default. SP_D becomes a copy of PV_D when the block's target mode is Man. A manually entered SP_D value is overwritten on the block's next execution cycle. This can prevent a state change when transitioning from Man to Auto mode. You can disable this option in OOS mode only.

SP-PV Track in LO or IMan – When this option is selected, SP_D becomes a copy of PV_D when the actual mode of the block is Initializing Manual or Local Override.

The block goes to IMan mode under the following conditions:

- No DO card or the wrong card is present.
- Communication with the H1 carrier is bad.
- The transducer block in the H1 carrier is in OOS mode.

The block goes to LO mode when the fault state is active.

Tgt to Man if Fault State Active – This option causes the target mode of the block to become Man when the fault state is active. Any of the following conditions causes the fault state to be active:

- The actual mode of the block is Cas and the status of CAS_IN_D is *BAD*.
- The actual mode of the block is Cas, and the substatus of CAS_IN_D is Initiate Fault State.
- The fault state is set manually in the resource block in the H1 carrier.

Use Fault st val on Restart – This option causes OUT_D to have the value of FSTATE_VAL_D on a restart (that is, when the carrier is powered up).

Fault State to value – This option causes OUT_D to have the value of FSTATE_VAL_D when the fault state is active. This occurs FSTATE_TIME seconds after the fault state is detected if the condition still exists. When the option is not selected, OUT_D keeps its value when the fault state is active.

Use PV for BKCAL_OUT – This option results in the value of BKCAL_OUT_D being that of PV_D. Normally BKCAL_OUT is the value of SP_D.

Invert – When this option is enabled (True), OUT_D becomes an inverted copy of SP_D. When this option is not enabled (False), OUT_D is a straight copy of SP_D. Inversion is done on a bit by bit basis.

Note You can set I/O options in Out of Service mode only.

Simulation

When SIMULATE_D is not enabled and the mode is not Out of Service (OOS), the value of OUT_D is sent to the hardware. When SIMULATE_D is enabled, the status of OUT_D is forced to *Good*, regardless of the condition of the hardware. In addition, READBACK_D and PV_D take on the value and status of SIMULATE_D.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Simulate active – SIMULATE_D is enabled; therefore, PV_D is not real.

Input failure/process variable has *Bad status* – The source of the block's process variable is bad. For the FFMDO block, this might indicate a hardware failure or a Bad status on the SIMULATE_D parameter.

Output failure – The output hardware is Bad.

Out of Service – The block is not being processed.

Modes - Multiple Discrete Output Function Block

The Multiple Discrete Output function block supports the following modes.

Initialization Manual (IMan)

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

Cascade (Cas)

Local Override (LO)

Remote Cascade (RCas)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Multiple Discrete Output Function Block

Under normal conditions, the output status of OUT_D is *Good: Non-Cascade* and the output status of BKCAL_OUT_D is *Good: Cascade*. However, the block supports a status action on failure and block error indications.

Action on Failure

When the output hardware goes bad, the OUT_D and BKCAL_OUT_D statuses are set to *Bad: DeviceFail* and the block is set to IMan mode. The BLOCK_ERR parameter shows Output failure. When SIMULATE_D is enabled, OUT_D does not show *Bad* status. When the fault state is active, the block goes to Local Override mode.

Parameters - Multiple Discrete Output Function Block

The following table lists the system parameters for the Multiple Discrete Output function block:

Multiple Discrete Output Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.

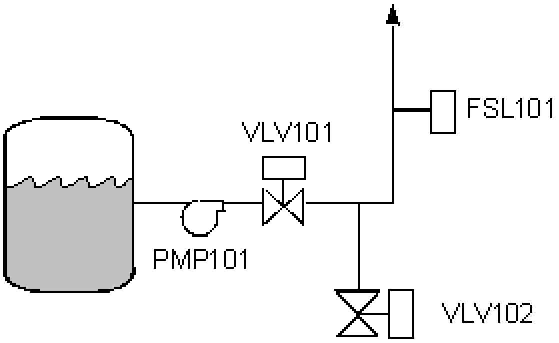
Parameter	Units	Description
BKCAL_OUT_D	None	The value and status required by the BKCAL_IN_D input of another block.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Multiple Discrete Output function block are Simulate active, Input failure/process variable has *Bad* status, Output failure, and Out of Service.
CAS_IN_D	None	The remote setpoint value from another block.
FSTATE_TIME	Seconds	The time in seconds from detection of fault of the output block remote setpoint to the output action of the block output if the condition still exists.
FSTATE_VAL_D	EU of PV	The preset discrete SP_D value to use when fault occurs. Ignored if the I/O option Fault State to value is false.
IO_OPTS	None	I/O options. Allow you to select how the I/O signals are processed. The supported I/O options for the Multiple Discrete Output function block are: Use PV for BKCAL_OUT, Tgt to Man if Fault State Active, Use fault st val on restart, Fault State to value, SP-PV Track in Man, SP-PV Track in LO or IMan, and Invert.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.
OUT_D	None	The 8-bit integer output value and status.
PV_D	None	The 8-bit integer process variable calculated from READBACK_D.
RCAS_IN_D	EU of SP_SCALE	The remote setpoint value and status. Input provided by a device or the output of another block.
RCAS_OUT_D	EU of PV_SCALE	The remote setpoint value and status output provided to a device to allow action to be taken under mode change.
READBACK_D	None	The value and status of OUT_D unless simulation is enabled, in which case it is the simulated value and status.
SHED_OPT	None	Defines action to be taken on remote control device time out.
SIMULATE_D	None	Enables simulation and allows you to enter an 8-bit integer value and status.
SP_D	None	The discrete target block output value (setpoint).
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.

Parameter	Units	Description
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The applicable status option for this block is Propagate Fault Backward. When this option is selected and the fault state is Active, the substatus on BKCAL_OUT is Fault State Active.
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Multiple Discrete Output and Input Function Blocks

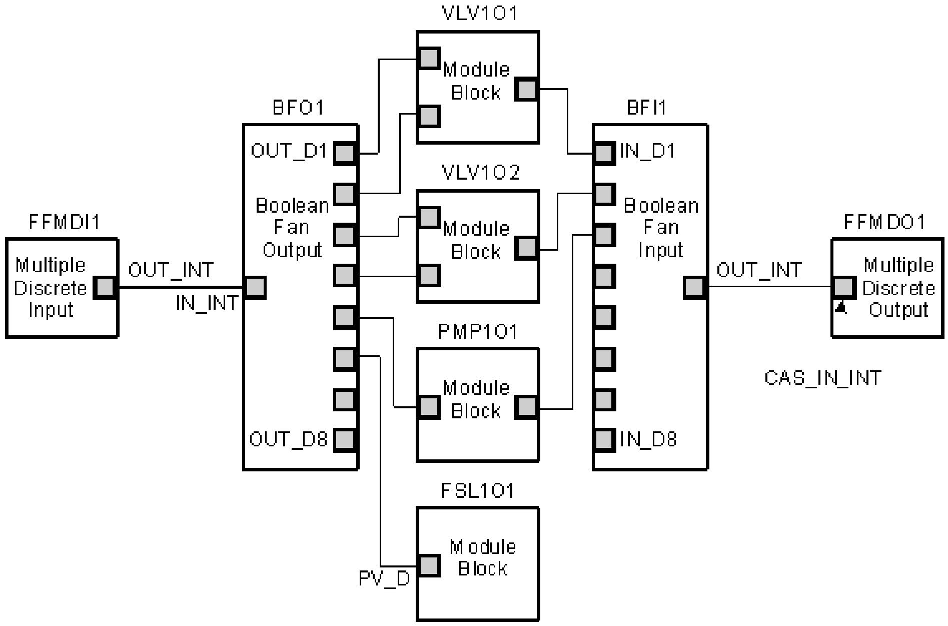
Both the Multiple Discrete Input (FFMDI) and Multiple Discrete Output (FFMDO) function blocks are ideal for controlling multiple discrete devices that are installed a relatively long distance from the controller. Because the blocks execute in the H1 carrier, the information can be transmitted through the H1 segment. In the following example, these function blocks are used to control and monitor a pump, two discrete valves and a flow switch.



Use Example for FFMDI and FFMDO

The FFMDI and FFMDO blocks are typically used in conjunction with the Boolean Fan Input (BFI) and Boolean Fan Output (BFO) function blocks. When the FFMDI block is assigned to an H1 carrier, the block is associated with the Discrete Input card connected to the carrier. When the FFMDO block is assigned to an H1 carrier, the block is associated with the Discrete Output card connected to the carrier. Boolean fan blocks can be used to isolate the values associated with the channels.

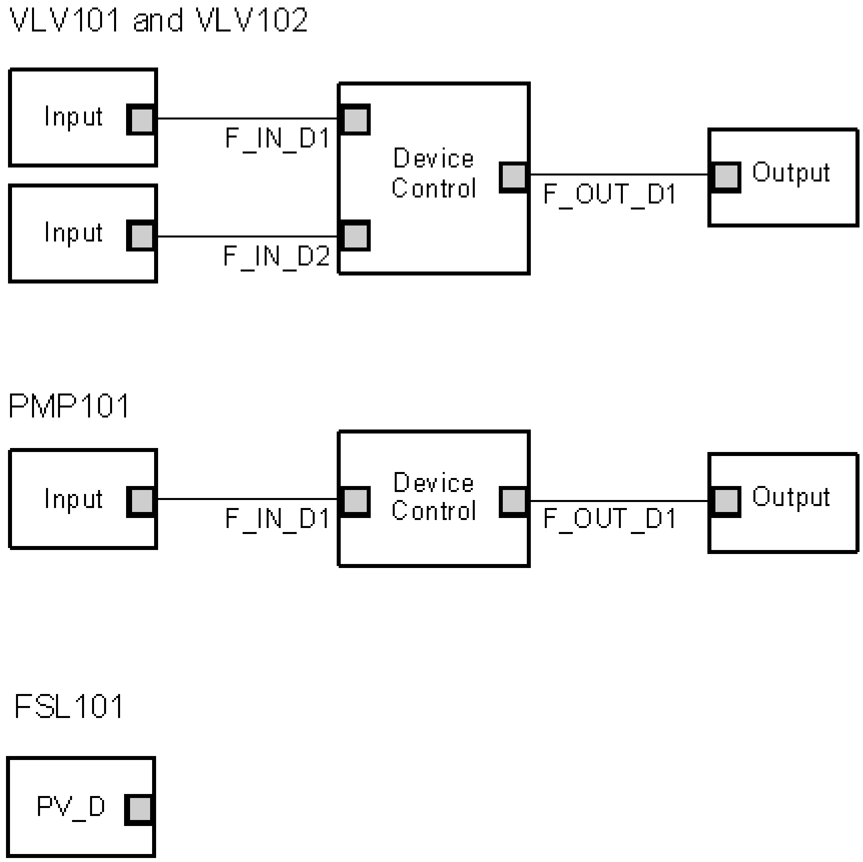
Caution One or more bad inputs to a BFI function block results in a bad output. This causes an FFMDO block to shed mode and hold outputs. In the application shown, bad inputs are not likely. Review your application to determine the potential for bad inputs to the BFI block and the possible consequences. Use the Calculation/Logic function block as an alternative. Also, refer to the sample code at the end of this application example.



Schematic for Use Example

In this example, module blocks VLV101, VLV102, and PMP101 are references to modules containing Device Control function blocks. The module block FSL101 is a reference to a module containing an input parameter used to generate an alarm for the flow switch.

The module blocks in the example have the following construction.



Schematic of Module Blocks for the Use Example

In the example, FFMDI1 reads the Discrete Input card channels. BFO1 distributes these values to the module blocks. The outputs of these blocks are brought into BFI1 and are then output to FFMDO1 to drive the channels on the Discrete Output card on the H1 carrier.

The input parameter in FSL101 contains a low flow alarm. When the flow is low, PV_D goes to zero, triggering an alarm.

Note that the input and output parameters in the module blocks must be Discrete with Status.

Sample Code for Calc/Logic Block Application

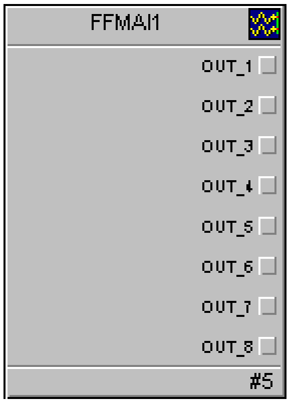
The following code is an example of how you can use the Calc/Logic block as an alternative to the Boolean Fan Input block for applications that use the Multiple Discrete Output function block.

```
OPTION EXPLICIT
 VAR
  VAL1; VAL2;
 END_VAR;
VAL1 := OUT1;
IF 'IN1.ST' = GOOD THEN
 IF IN1 = TRUE THEN
  VAL1 := VAL1 | 1;
 ELSE
  VAL1 := VAL1 & 254;
 ENDIF;
ENDIF;
IF 'IN2.ST' = GOOD THEN
 IF IN2 = TRUE THEN
  VAL1 := VAL1 | 2;
 ELSE
  VAL1 := VAL1 & 253;
 ENDIF;
ENDIF;
IF 'IN3.ST' = GOOD THEN
 IF IN3 = TRUE THEN
  VAL1 := VAL1 | 4;
 ELSE  VAL1 := VAL1 & 251;
 ENDIF;
ENDIF;
IF 'IN4.ST' = GOOD THEN
 IF IN4 = TRUE THEN
  VAL1 := VAL1 | 8;
 ELSE
  VAL1 := VAL1 & 247;
 ENDIF;
ENDIF;
IF 'IN5.ST' = GOOD THEN
 IF IN5 = TRUE THEN
  VAL1 := VAL1 | 16;
 ELSE
  VAL1 := VAL1 & 239;
 ENDIF;
ENDIF;
IF 'IN6.ST' = GOOD THEN
 IF IN6 = TRUE THEN
  VAL1 := VAL1 | 32;
 ELSE
  VAL1 := VAL1 & 223;
 ENDIF;
ENDIF;
IF 'IN7.ST' = GOOD THEN
 IF IN7 = TRUE THEN
```

```
  VAL1 := VAL1 | 64;
 ELSE
  VAL1 := VAL1 & 191;
 ENDIF;
ENDIF;
IF 'IN8.ST' = GOOD THEN
 IF IN8 = TRUE THEN
  VAL1 := VAL1 | 128; ELSE  VAL1 := VAL1 & 127; ENDIF;ENDIF;OUT1 := VAL1;
```

Multiplexed Analog Input (FFMAI) Function Block

The MAI function block can process up to eight fieldbus device measurements from a single device. This function block is for fieldbus applications where the sensor types are the same. The MAI block can optimize the communications on an H1 segment by providing all of the information for up to 8 measurements in a single function block.

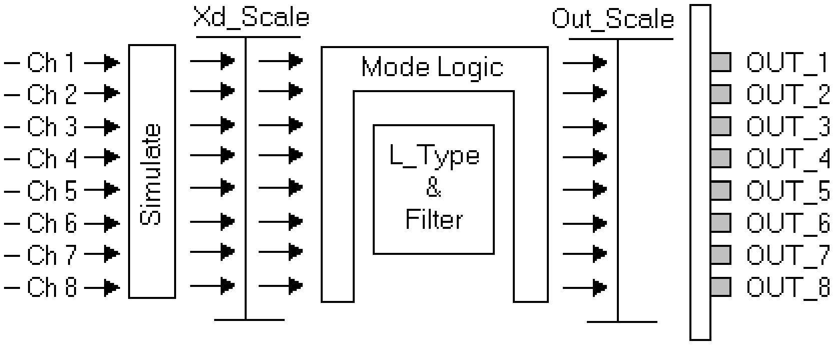


Multiplexed Analog Input Function Block

Note that the MAI block uses many of the same parameters to support all eight inputs. One consequence of this is that the RTD’s or thermocouples connected to an MAI block must all have the same range. In a similar way, the simulation value and mode handling applies to all outputs as well. The MAI block is ideal for higher density fieldbus transmitters, such as the Rosemount Model 848T transmitter.

Schematic Diagram - Multiplexed Analog Input Function Block

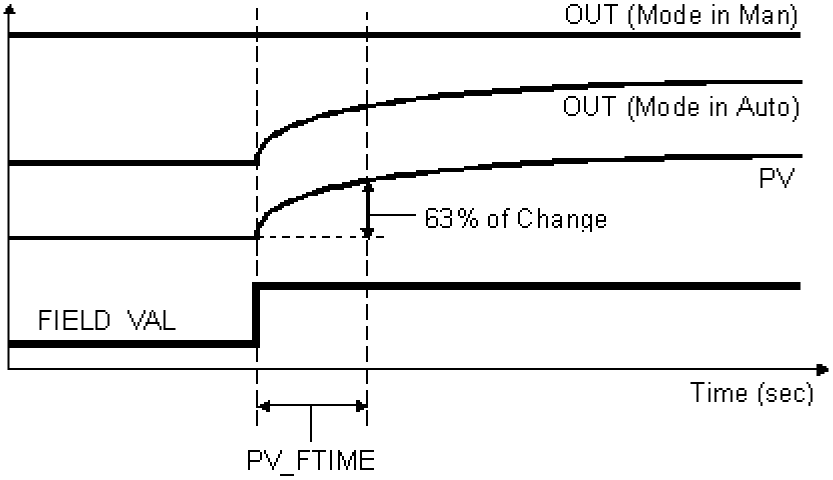
The following diagram shows the internal components of the Multiplexed Analog Input function block:



Multiplexed Analog Input Function Block Schematic Diagram

Block Execution - Multiplexed Analog Input Function Block

The following diagram shows the timed response of the Multiplexed Analog Input function block:



Multiplexed Input Function Block Timing Diagram

You select the manner of processing the analog measurement value by configuring signal conversion and filtering parameters.

Signal Conversion

You choose direct, indirect, or indirect square root signal conversion with the linearization type parameter (L_TYPE).

The block applies transducer scaling (XD_SCALE) to the value from the channel to produce the FIELD_VAL in percent. The XD_SCALE units code must match the channel units code (if one exists), or the block remains in OOS mode after configuration. The OUT_SCALE is normally the same as the transducer, but if L_TYPE is set to *Indirect* or *Ind Sqr Root*, OUT_SCALE determines the conversion from FIELD_VAL to the output. PV and OUT always have identical scaling. OUT_SCALE provides scaling for PV. The PV is always the value that the block puts in OUT if the mode is Auto. If Man is allowed, someone may write a value to the output. The status prevents any attempt at closed loop control using the Man value by setting the Limit value to Constant.

Equations:FIELD_VAL = 100 * (channel value - EU @ 0%) / (EU @ 100% - EU @ 0%) [XD_SCALE]
Direct: PV = channel value
Indirect: PV = (FIELD_VAL/100) * (EU @ 100% - EU @ 0%) + EU @ 0% [OUT_SCALE]
Ind Sqr Root: PV = sqrt(FIELD_VAL/100) * (EU @ 100% - EU @ 0%) + EU @ 0% [OUT_SCALE]

Select one of the following signal conditioning options:

Direct signal conditioning – simply passes through the accessed channel input value (or the simulated value when simulation is enabled).

Indirect signal conditioning – linearly converts the accessed channel input value (or the simulated value when simulation is enabled) from its specified range (XD_SCALE) to the range and units of the PV and OUT parameters (OUT_SCALE).

Indirect square root signal conditioning – converts the accessed channel input value (or the simulated value when simulation is enabled) from its specified range (XD_SCALE) by taking the square root of the value and scaling it to the range and units of the PV and OUT parameters (OUT_SCALE).

You view the accessed value (in percent of XD_SCALE) through the FIELD_VAL parameter.

When the converted input value is below the limit specified by the LOW_CUT parameter and the Low Cutoff I/O option (IO_OPTS) is enabled (True), a value of zero is used for the converted value (PV). This option can be useful with zero-based measurement devices, such as flowmeters.

Note You can set the I/O option in Out of Service mode only.

Filtering

You apply filtering to the converted value (PV) by specifying the filter time constant (in seconds) in the PV_FTIME parameter. When you specify a value of zero, no filtering is applied.

Block Errors

The following 16 conditions can be reported in the BLOCK_ERR parameter:

- **Other**
- **Block Configuration Error**: the selected channel carries a measurement that is incompatible with the engineering units selected in XD_SCALE, the L_TYPE parameter is not configured, or CHANNEL = zero.
- **Link Configuration Error**
- **Simulate Active**: Simulation is enabled and the block is using a simulated value in its execution.
- **Local Override**
- **Device Fault State Set**
- **Device Needs Maintenance Soon**
- **Input Failure/Process Variable has *Bad Status***: The hardware is bad, or a bad status is being simulated.
- **Output Failure**: The output is bad based primarily upon a bad input.
- **Memory Failure**
- **Lost Static Data**
- **Lost NV Data**
- **Readback Failed**

- **Device Needs Maintenance Now**
- **Power Up**
- **Out of Service**: The actual mode is Out of Service.

Modes - Multiplexed Analog Input Function Block

The Multiplexed Analog Input function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of a block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Multiplexed Analog Input Function Block

The STATUS_OPTS parameter reflects the measurement value and the operating condition of the channel in the Fieldbus device.

There are four MAI block status options:

- Uncertain if Man mode -- Sets the OUT_*n* status to Uncertain if the block is in Manual mode.
- Bad if Limited -- Sets the OUT_*n* status to Bad if the channel input status is limited.
- Uncertain if Limited -- Sets the OUT_*n* status to Uncertain if the channel input status is limited.
- Propagate Fault Forward -- If the status from the sensor is Bad - Device Failure or Bad - Sensor Failure, propagate it to OUT_*n* without generating an alarm. The use of these substatuses in OUT_*n* is determined by this option. Through this option, you can determine whether alarming (sending of an alert) will be done by the block or propagated downstream for alarming.

Note You can set the status option (STATUS_OPTS) in Out of Service mode only.

In Auto mode, OUT_*n* reflects the value and status quality of the measurement. In Man mode, the OUT_*n* status high and low limits are set to indicate that the value is a constant and the OUT_*n* status is always Good.

Parameters - Multiplexed Analog Input Function Block

The following table lists the parameters for the Multiplexed Analog Input function block.

Multiplexed Analog Input Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block.
CHANNEL	None	Allows for custom channel setting. Valid values include: - 0 = Uninitialized - 1 = Channels 1 to 8 can only be set to their corresponding channel number - 2 = Custom settings. Channels can be configured for any valid channel defined in the device description.
CHANNEL_*n* where *n* = 1 through 8	None	The number of the logical hardware channel that is connected to the I/O block. It defines the transducer to be used going to or from the physical world.
IO_OPTS	None	I/O options. Allow you to select how the I/O signals are processed. The supported I/O option for the Multiplexed Analog Input function block is Low Cutoff.
L_TYPE	None	Linearization type. Determines whether the field value is used directly (Direct), is converted linearly (Indirect), or is converted with the square root (Indirect Square Root).

Parameter	Units	Description
LOW_CUT	Percent	Activated when the Low Cutoff I/O option is enabled (True). When the converted measurement is below the LOW_CUT value, the PV is set to 0.0.
MODE	None	Parameter used to request and show the source of the output used by the block.
OUT_*n* where *n* = 1 through 8	EU of OUT_SCALE	The primary value and status calculated by the block in Auto mode. OUT can be set manually in Man mode.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
PV_FTIME	Seconds	The time constant of the first-order PV filter. It is the time required for a 63% change in the FIELD_VAL to be reflected in the PV.
SIMULATE	None	Enables simulation and allows you to enter an input value and status. The SIMULATE value is used by the block only when SIMULATE is enabled.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The supported status options for the Multiplexed Analog Input function block are: Uncertain if Man mode BAD if Limited Uncertain if Limited Propagate Fault Forward
STRATEGY	None	Use the strategy field to group blocks. This data is not checked or processed by the block.
SUBSTITUTE_IN	Determined by source or EU of PV_SCALE	If there is a problem communicating with the field device and SUBSTITUTE_IN has a non-Bad status, the controller passes the SUBSTITUTE_IN value through to the output parameter(s) of the shadow block.*
XD_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the channel input value.

Application Information - Multiplexed Analog Input Function Block

Note When the range or units of the transmitter's primary variable for your application is different than that defined in OUT_SCALE, set L_TYPE to Indirect and define the input and output ranges in XD_SCALE and OUT_SCALE.

Application Example

This function block is for fieldbus applications where the sensor types and functionality of each channel (that is, values for simulate, scaling, filtering, mode, and so on) are the same. The MAI block can optimize the communications on an H1 segment by providing all of the information for the 8 measurements in a single function block.

Use the MAI block when higher density Fieldbus transmitters are being used. The Rosemount Model 848T device is a typical example of a higher density Fieldbus transmitter. The Model 848T supports up to eight Thermocouples or RTDs. You can configure the eight signals using a combination of AI function blocks and the MAI function block.

The MAI block and AI blocks can both be used within one device. The MAI block can process the eight measurements in the device or be set to custom and have the measurements selected. The AI block can process an individually selected measurement, even one already being processed in the MAI block.

Note that the MAI block uses the same scaling parameters (L_TYPE, XD_SCALE, OUT_SCALE) for all eight inputs. This means that the RTD’s or thermocouples must all have the same range in order to use the same MAI block. Also, the same simulation value and mode handling applies to all outputs.

The MAI block does not have built-in standard alarms. Use an alarm block to generate standard alarms for an individual channel from the MAI block. If separate tag names (module names) are required for each channel the Alarm Module Template (found in the DeltaV library Module Templates/Monitoring) can be used.

Alarms generated by a Fieldbus device can be viewed in the Event Chronicle and DeltaV Operate. The unit and area assignment for a Fieldbus device is determined by the lowest object index in the device. This is typically the primary measurement or output of a fieldbus device. For the Model 848T, this means that the module assigned to the Analog Input function block object index 2400 is the module used to determine the area to which the alarms from the Model 848T will be assigned. If there is no module assigned to this function block, this device will be assigned to the same area as it’s controller (which defaults to area A).

Example:

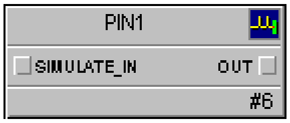
On the Model 848T, you have two temperature measurements with unique ranges and 6 with the same ranges. You could use two AI blocks for the unique ranges and 1 MAI block for the other 6 measurements.

In the above example, suppose each of the six measurements need separate tag indication (TI-1 through TI-6) along with each measurement having a HIHI and a LOLO alarm. This could be accomplished as follows:

1. Create a module TI-0 using the MAI block. This module communicates with the Model 848T and contains the information from the MAI block.
2. Create six additional modules named TI-1 through TI-6 using the module template (found in the DeltaV library Module Templates/Monitoring) ALARM. Each of these modules will provide alarming of one of the MAI outputs.
3. Edit each of the six alarm modules, assigning the external reference parameter to one of the channels in the MAI block. Configure the HIHI and LOLO alarms.

Pulse Input (PIN) Function Block

The Pulse Input (PIN) function block is an I/O block used to provide analog input values from Pulse Input channels on the Multifunction I/O card.



Pulse Input Function Block

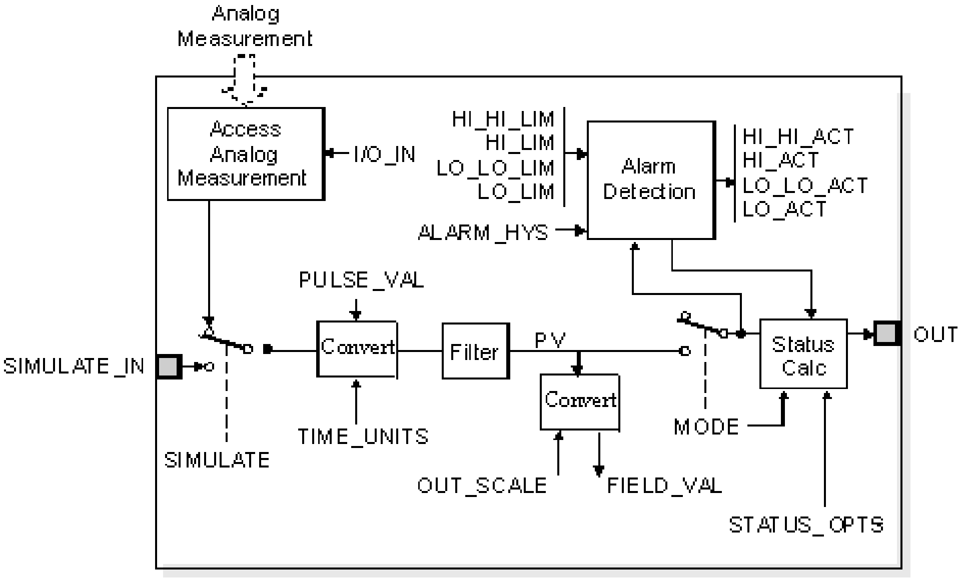
The PIN function block supports block alarming, signal scaling, signal filtering, signal status calculation, mode control, and simulation.

In Automatic mode, the block's output parameter (OUT) reflects the process variable (PV) value and status. In Manual mode, OUT can be set manually.

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block through the SIMULATE_IN input.

Schematic Diagram- Pulse Input Function Block

The following diagram shows the internal components of the Pulse Input function block:



Pulse Input Function Block Schematic Diagram

Block Execution - Pulse Input Function Block

You select the manner of processing the pulse value by configuring the I/O selection and filtering parameters.

I/O Selection

When you configure the Pulse Input function block, you select the I/O channel associated with an pulse measurement by configuring the Device Signal Tag (DST) of the IO_IN parameter. You select the Device Tag and the parameter the PIN block accesses on that channel.

The valid channel parameter for IO_IN is:

FREQUENCY – The frequency of pulses on a pulse input channel (counts per second)

Simulation

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block.

During configuration, decide whether you want to enter the simulated value/status manually during operation or use a value/status from another block for the simulated value/status.

When the value is entered manually:

- The operator first enables simulation by selecting the SIMULATE parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field. In online operation, the operator can enter a simulated status value in the Simulate Status field.
- If SIMULATE_IN is not connected (status = *Bad: NotConnected*), the SIMULATE value is used.
- If SIMULATE_IN has any status except *Bad: NotConnected*, the SIMULATE_IN value is used. In this case, the SIMULATE_IN value always overrides the SIMULATE value.

Note Use the SIMULATE parameter to enter values manually and make sure SIMULATE_IN is not connected. If SIMULATE_IN is connected or a value is entered manually into SIMULATE_IN, the SIMULATE_IN value overrides the SIMULATE value.

When the value/status from another block is used:

- During configuration, connect SIMULATE_IN to the desired block output or parameter. Do not enter a value in the Simulate Value field of the SIMULATE_IN input; the block uses the connected value automatically.
- During operation, the operator enables simulation by selecting the SIMULATE parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.

Note Do not enter a value for the SIMULATE_IN parameter. If you do and the status of SIMULATE_IN is not *Bad: NotConnected*, the manually entered value for SIMULATE_IN overrides any value you enter in SIMULATE.

Filtering

You apply filtering to the converted value (PV) by specifying the filter time constant (in seconds) in the PV_FTIME parameter. When you specify a value of zero, no filtering is applied.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Simulate active – Simulation is enabled and the block is using a simulated value in its execution.

Input failure/process variable has *Bad status* – The source of the block's process variable is bad. Indicates a hardware failure, a non-existent Device Signal Tag (DST), or a Bad status on the SIMULATE parameter.

Out of Service – The block is in Out of Service (OOS) mode.

Calculation of OUT

OUT equals PV and status when in Auto mode. The operator sets OUT when in Manual mode.

The DeltaV Multifunction card calculates the frequency of pulses in counts per second. Therefore, the block does not use scan rate in the calculation of OUT.

Calculation of PV

The Pulse Input function block's PV is calculated by frequency *times* pulse value *times* time units factor.

The DeltaV Multifunction card calculates the frequency of pulses in counts per second. Therefore, the block does not use scan rate in the calculation of PV.

Calculation of FIELD_VAL

The Pulse Input function block's FIELD_VAL is calculated as the percent value of PV within the range defined by OUT_SCALE.

Modes - Pulse Input Function Block

The Pulse Input function block supports these modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of a block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Alarms - Pulse Input Function Block

Block alarm detection is based on the OUT value. You can configure the alarm limits of the following standard alarms:

- High (HI_LIM)
- High high (HI_HI_LIM)
- Low (LO_LIM)
- Low low (LO_LO_LIM)

Status Handling - Pulse Input Function Block

Normally, the status of the PV reflects the measurement value, the operating condition of the I/O card, and any active alarm condition.

In Auto mode, OUT reflects the value and status quality of the PV. In Man mode, the OUT status high and low limits are set to indicate that the value is a constant and the OUT status is always *Good*.

Parameters - Pulse Input Function Block

The following table lists the system parameters for the Pulse Input function block:

Pulse Input Function Block System Parameters

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Pulse Input Function Block

The PIN block can be used with the Multifunction card PCI channel to obtain a flow rate to use in control. For example, suppose that you have a flow measurement device that represents the flow with a pulsed signal. Configure the PIN block to use the PCI channel's Frequency parameter (pulses in counts per second) to produce an analog floating point value based on the pulse rate which the PID block uses for control. Remember to set the PULSE_VAL and TIME_UNITS parameters to obtain the correct value on the output (EU/TIME_UNITS format).

Analog Control Blocks

This chapter contains information on analog control function blocks in the DeltaV system.

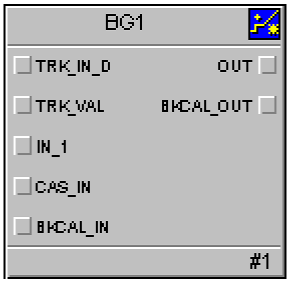
Bias/Gain Function Block

The Bias/Gain (BG) function block provides adjustable gain capability by computing an output value from a bias setpoint, an input, and a gain value. The block supports output tracking. The Bias/Gain function block is designed to be used as an element in parallel control paths, such as a plant master driving multiple boiler masters.

In Automatic (Auto) mode, the output is computed automatically and the bias setpoint can be adjusted manually. In Manual (Man) mode, the output can be set manually.

To provide bumpless transfer and anti-reset windup, you connect the BKCAL_OUT parameter to a master controller and the BKCAL_IN parameter to a slave controller. When bumpless transfer is not achieved completely by BKCAL_OUT and BKCAL_IN, an internal bias is computed and applied at transfer time. This bias is ramped over a specified time period.

You connect the tracking inputs (TRK_IN_D and TRK_VAL) for externally controlled output tracking.



Bias/Gain Function Block

IN_1 is the input value and status.

BKCAL_IN is the analog input value and status from another block's BKCAL_OUT output that is used for backward output tracking.

TRK_IN_D is the Boolean input used to initiate the external tracking function.

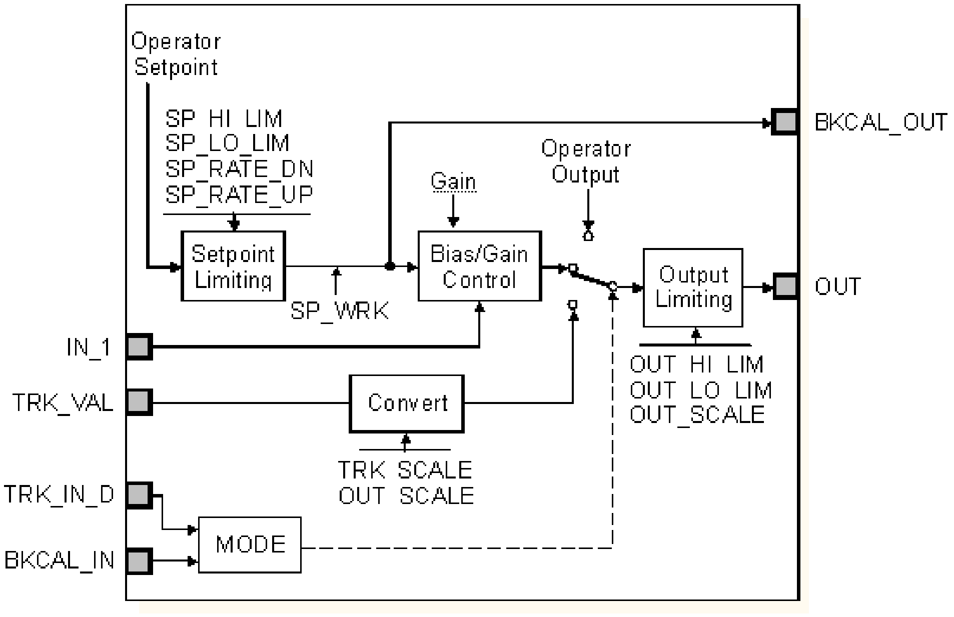
TRK_VAL is the input value OUT is set to in Local Override (LO) mode (while TRK_IN_D is true).

OUT is the biased analog output value and status.

BKCAL_OUT is the output value connected to the BKCAL_IN input of another block to prevent reset windup and to provide bumpless transfer to closed loop control.

Schematic Diagram - Bias/Gain Function Block

The following diagram shows the internal components of the Bias/Gain function block:



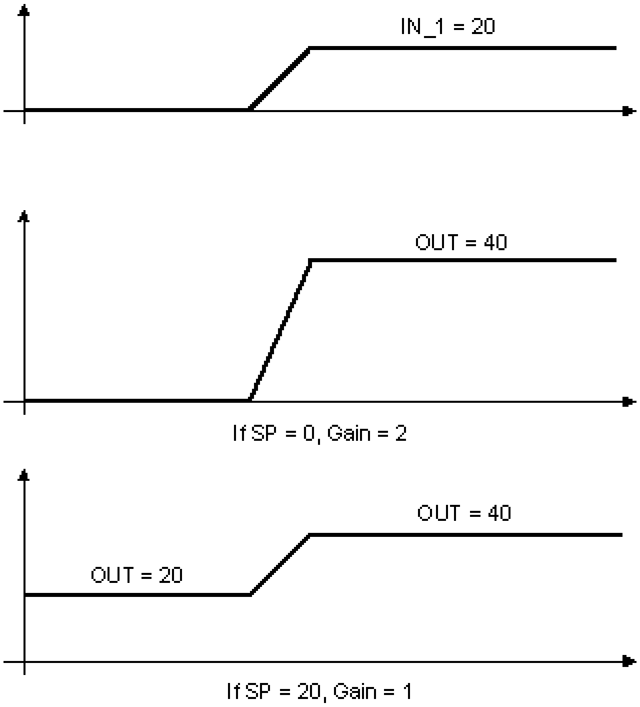
Bias/Gain Function Block Schematic Diagram

Block Execution - Bias/Gain Function Block

The block computes the output value (OUT) as follows:

$$OUT = (SP + IN\_1) \times GAIN$$

There are two ways to modify the input. Either offset the input from its original value by a specific amount or scale the input. The following figure illustrates each of these cases for a single input. The figure shows the two methods of modification separately. However, these methods can be combined so that Bias/Gain delivers scaling and offset together.



Bias/Gain Control Signal Modification Diagram

SP offsets the input by a given amount, and GAIN scales the modified input. In the first example in the above figure, SP is 0, and OUT is GAIN * IN_1. However, in the second example, SP has a non-zero value, and GAIN is set to 1 (no scaling). Therefore, OUT is IN_1 + SP.

Note To pass the input directly through unchanged, set SP = 0 and GAIN = 1.

Tracking

You specify output tracking with control options and parameters.

The Track Enable control option (CONTROL_OPTS) must be True for the track function to operate. When the Track in Manual control option is True, tracking can be activated and maintained when the block is in Man mode. When Track in Manual is False, the operator might override the tracking function when the block is in Man mode. Activating the track function causes the block's actual mode to go to Local Override (LO). You can set control options in Manual or Out of Service mode only.

The tracking value parameter (TRK_VAL) specifies the value to be converted and tracked into the output when the track function is operating. The tracking scale parameter (TRK_SCALE) specifies the range of TRK_VAL.

When the track control parameter (TRK_IN_D) is True and the Track Enable control option is True, the TRK_VAL input is converted to the appropriate value and output in units of OUT_SCALE.

Setpoint Limiting

You limit the setpoint by configuring the SP_HI_LIM and SP_LO_LIM parameters. You limit the rate of change of the setpoint by configuring the SP_RATE_UP and SP_RATE_DN parameters. The value of the setpoint after limiting is shown in the working setpoint (SP_WRK) parameter.

Output Limit Constraints

As part of download the output high and low limits are set to the configured values. If these limits have not been configured OUT_HI_LIM will be set to OUT_SCALE_HI and OUT_LO_LIM will be set to OUT_SCALE_LO.

The following constraints apply as part of download or on-line entry:

- OUT_HI_LIM is restricted to OUT_SCALE_HI+.1*(OUT_SCALE_HI-OUT_SCALE_LO).
- OUT_LO_LIM is restricted to OUT_SCALE_LO-.1*(OUT_SCALE_HI-OUT_SCALE_LO).

If a new scale entry causes a limit to be outside of these rules, the DeltaV system forces the limit within the rules.

The OUT parameter is not immediately changed as a result of changing the scale or limits. However, if OUT violates the new limits, the algorithm will change OUT on the subsequent execution in order to satisfy the new limits if required.

Output Selection and Limiting

Output selection is determined by mode. In an automatic control mode, the output is computed by bias/gain control. In Man mode, the output can be entered manually. In LO mode the output is determined by TRK_VAL and its scaling.

You limit the output by configuring the OUT_HI_LIM and OUT_LO_LIM parameters.

Bumpless Transfer on Mode Transitions

The BAL_TIME parameter determines bumpless transfer operations. When the mode transitions from a non-automatic mode to an automatic mode, an internal balancing bias is calculated that maintains OUT. The internal bias is then ramped to zero over a period of time specified by BAL_TIME. When BAL_TIME is zero, the internal bias and ramp to 0 (zero) are not applied. When BAL_TIME is non-zero, the internal bias and ramp to 0 are applied.

When ACT_ON_IR in CONTROL_OPTS is chosen the setpoint is back calculated for bumpless transfer on transition from a non-automatic to an automatic mode. Since the SP could be outside the SP limits, and therefore a bump is possible, it is wise to have a non-zero BAL_TIME to cushion or ramp the bump as described in the previous paragraph.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Local Override – The block is in Local Override (LO) mode.

Out of Service – The block is in Out of Service (OOS) mode.

Modes - Bias/Gain Function Block

The Bias/Gain function block supports seven modes:

Out of Service (OOS)

Initialization Manual (IMan)

Local Override (LO)

Manual (Man)

Automatic (Auto)

Cascade (Cas)

Remote Cascade (RCas)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Bias/Gain Function Block

When the IN_1 status is evaluated Bad, the block will drop to an actual mode of MAN. Other status behavior is dependent upon the settings in the parameter STATUS_OPTS.

Supported STATUS_OPTS

Use Uncertain as Good – If this option is selected, a status of Uncertain for IN_1 will be treated as good. If it is not selected, Uncertain will be treated as Bad, resulting in the block's dropping to MAN mode. – If this option is selected, a status of Uncertain for IN_1 will be treated as good. If it is not selected, Uncertain will be treated as Bad, resulting in the block's dropping to MAN mode.

IFS if Bad CAS_IN – (only supported in fieldbus devices) If CAS_IN status is Bad, OUT substatus is set to IFS (Initiate Fault State) and, thereby, ispropagated to a downstream block.(only supported in fieldbus devices) If CAS_IN status is Bad, OUT substatus is set to IFS (Initiate Fault State) and, thereby, ispropagated to a downstream block.

IFS if Bad IN – (only supported in fieldbus devices) If IN_1 status is Bad, OUT substatus is set to IFS (Initiate Fault State) and, thereby, is propagated to a downstream block.(only supported in fieldbus devices) If IN_1 status is Bad, OUT substatus is set to IFS (Initiate Fault State) and, thereby, is propagated to a downstream block.

Target to Manual if Bad IN – If IN_1 status is evaluated as Bad, the target mode drops to MAN.If IN_1 status is evaluated as Bad, the target mode drops to MAN.

Note You can adjust STATUS_OPTS in Manual or Out of Service mode only.

Parameters - Bias/Gain Function Block

The following table lists the system parameters for the Bias/Gain function block:

Bias/Gain Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for dissipation of the internal balancing bias.

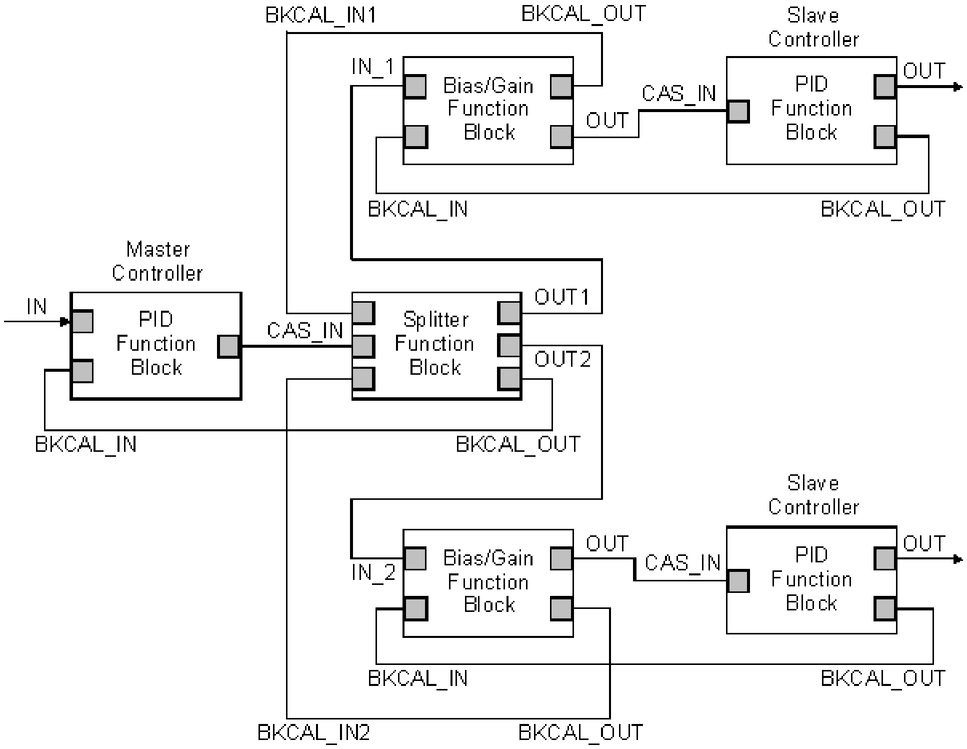
Parameter	**Units**	**Description**
BKCAL_IN	EU of OUT_SCALE	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BKCAL_OUT	EU of OUT_SCALE	The value and status required by the BKCAL_IN input of another block to prevent reset windup and to provide bumpless transfer to closed loop control.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Bias/ Gain function block are Local override and Out of Service.
CAS_IN	EU of PV_SCALE	The remote analog setpoint value from another block.
CONTROL_OPTS	None	Control options. Allow you to specify control strategy options. The supported control options for the Bias/Gain function block are No OUT Limits in Manual, Obey SP Limits if Cas or RCas, Use BKCAL_OUT With IN_1, Act on IR, Track in Manual, Track Enable, and SP Track Retained Target.
FRSIRB_OPTS	None	FRSI RB options. The supported option is Treat_IN_1 As Wild. When Treat_IN_1 As Wild is true regardless of cascade or non-cascade connection IN_1 is treated as a non-cascade input and if Use_Bkcal_out_With_IN_1 is true Bkcal_out sub-status will be set to keep cascade closed.
GAIN	None	The multiplier applied in GAIN * (SP + IN_1).
IN_1	Determined by source	The input with status to which the bias (SP) is applied.
MODE	None	Parameter used to show and set the block operating state.
OUT	EU of OUT_SCALE	The block output value and status.
OUT_HI_LIM	EU of OUT_SCALE	The maximum output value allowed.
OUT_LO_LIM	EU of OUT_SCALE	The minimum output value allowed.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
RCAS_IN	EU of SP_SCALE	The remote setpoint value and status. Input provided by a remote device or the output of another block.
RCAS_OUT	EU of PV_SCALE	The output provided for bumpless mode transfer and reset limiting by the source of RCAS_IN. Essentially the equivalent of BKCAL_OUT for RCAS_IN.
SHED_OPT	None	Defines action to be taken on remote control device time out.*
SP	EU of OUT_SCALE	The block's setpoint value.

Parameter	Units	Description
SP_HI_LIM	EU of OUT_SCALE	The highest setpoint value allowed.
SP_LO_LIM	EU of OUT_SCALE	The lowest setpoint value allowed.
SP_RATE_DN	EU of OUT_SCALE per second	Rate of change limit for decreasing setpoint. When SP change violates the limit the effect is a ramp into the working setpoint (SP_WRK).
SP_RATE_UP	EU of OUT_SCALE per second	Rate of change limit for increasing setpoint. When SP change violates the limit the effect is a ramp into the working setpoint (SP_WRK).
SP_WRK	EU of OUT_SCALE	The working setpoint of the block. It is the result of setpoint limiting and setpoint rate of change limiting.
STATUS_OPTS	None	Status options. Allows you to select options for status handling and processing. The supported status options are Target to Manual if Bad IN and Use Uncertain as Good.
TRK_IN_D	None	Discrete input that initiates external tracking.
TRK_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the external tracking value (TRK_VAL).
TRK_VAL	EU of TRK_SCALE	The input value with status applied to OUT in LO mode.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Bias/Gain Function Block

The Bias/Gain function block is often used as a slave to a Splitter function block which is, in turn, a slave to the master controller in a control algorithm. The following diagram shows a typical example:

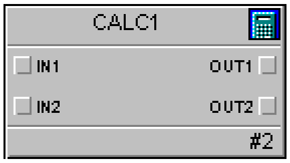


Bias/Gain Function Block Diagram Example

The master controller OUT parameter determines the setpoints of two slave PID blocks through a splitter and two BIAS/GAIN blocks. The Bias/Gain blocks allow each setpoint to be modified so that parallel field devices (for example, valves, pumps, fans) can be properly synchronized to the demands of a single master. Suppose the two slave PIDs control the speed of two variable-speed fans supplying a furnace. In this situation, the master controller monitors the total air demand and its OUT sets the total demand. The BIAS/GAIN blocks provide the means to apply more or less of the demanded control action through one or the other of the fans (by SP or GAIN adjustment).

Calculation/Logic Function Block

The Calculation/Logic (CALC) function block allows you to specify an expression that determines the block's output. Mathematical functions, logical operators, constants, parameter references, and I/O reference values can be used in the expression. There are no modes or alarm detection in the Calculation/Logic function block.



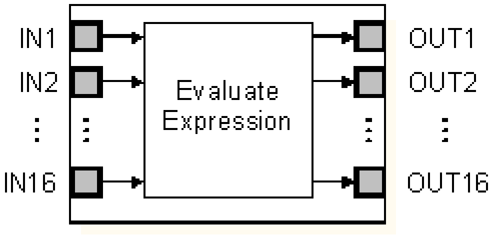
Calculation/Logic Function Block

IN1 through IN[n] are the inputs to the block (as many as 16 inputs).

OUT1 through OUT[n] are the block outputs (as many as 16 outputs).

Schematic Diagram - Calculation/Logic Function Block

The following diagram shows the internal components of the Calculation/Logic function block:



Calculation/Logic Function Block Schematic Diagram

Block Execution - Calculation/Logic Function Block

The Calculation/Logic function block uses as many as 16 inputs and 16 outputs to evaluate its contained expression. In addition, the expression evaluator uses constants and external references that you specify to evaluate the expression. The calculated values are assigned to external references or outputs for use as parameters or inputs to the control strategy in other blocks.

The number of inputs and outputs in the Calculation/Logic function block are extensible parameters. The block default is two inputs and two outputs. You add inputs or outputs by selecting the function block diagram and clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs and/or outputs. This creates additional input/output connectors for the block.

Expressions

Expressions are structured text in a specific syntax and are made up of operands, operators, functions, constants, and keywords. You write expressions using the Expression Editor.

Refer to the Expressions topic for syntax guidelines for writing expressions and information on the supported operands, operators, functions, constants, and keywords.

The Calculation/Logic function block supports IF-THEN-ELSE-END_IF structures.

When the IF {Expression} evaluates to True or non-zero, any commands following IF and prior to ELSE or END_IF (whichever occurs first) are executed.

When the IF {Expression} is False and ELSE is included, any commands after ELSE and before END_IF are executed.

When the IF {Expression} is False and ELSE is not included, all commands between IF and END_IF are ignored. In this case, program execution continues with the first command following END_IF.

You can nest an IF ... END_IF block within another IF ... END_IF block. Comments can be placed on the same line after IF, ELSE, and END_IF. Multiple statements can be placed between the THEN and ELSE keywords, as well as between the ELSE and ENDIF keywords.

In the following example, the condition tests whether '/Block1.mode.ACTUAL' is equal to manual. Notice that the '=' operator is not used as an assignment operator; it tests the two operands for equality. If the condition is True, the '/Block1.mode.TARGET' is set to AUTO; otherwise, OUT1 is set to the value of IN1.

```
IF '/Block1.mode.ACTUAL' = MAN THEN
'/Block1.mode.TARGET' := AUTO;
ELSE
OUT1 := IN1;
ENDIF;
```

Note In the Calc/Logic block, the OUT(n) and IN(n) parameters refer to the OUT(n).CV and IN(n).CV fields respectively.

It is not always necessary to use the ELSE portion of the statement. For example, if you want to set the MERROR parameter to True when the PV of LIC549 goes above 75, you write the expression as follows:

```
IF '//LIC549/PV.CV' > 75 THEN
'//LIC549/MERROR.CV' := TRUE;
ENDIF;
```

Note The CV extension in this example stands for current value. Some parameters also support a status value, ST.

If the Calculation Logic function block executes a divide by zero expression, the only effect is that status of the output is set to BAD.

Parameters - Calculation/Logic Function Block

The following table lists the system parameters for the Calculation/Logic function block:

Calculation/Logic Function Block System Parameters:

Parameter	Units	Description
IN1 to IN16	Determined by source	The analog input value and status. The number of inputs is an extensible parameter.
OUT1 to OUT16	Determined by expression	The analog output value and status. The number of outputs is an extensible parameter.
ABNORM_ ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALGO_OPTS	None	Algorithm options. When selected, the expression algorithm aborts after a read error. The Algorithm option is AbortOnReadErrors.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Calculation/Logic function block are Block configuration error, Output Failure, and Read Back Failure.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Calculation/Logic Function Block

You can use the Calculation/Logic function block for a wide variety of applications, including complex calculations, signal conditioning, or functionality not available in standard function blocks.

Application Example: Signal Correction

Assume you need to apply the following nonlinear equation to an input value to calculate a corrected signal:

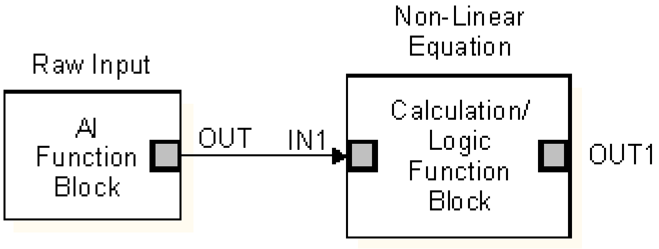
y = 3*x^3 + 2*x^2 + x + 1

Where:

y = corrected signal

x = raw input signal

This is an ideal application for the Calculation/Logic block. The following figure is the function block diagram for this application.



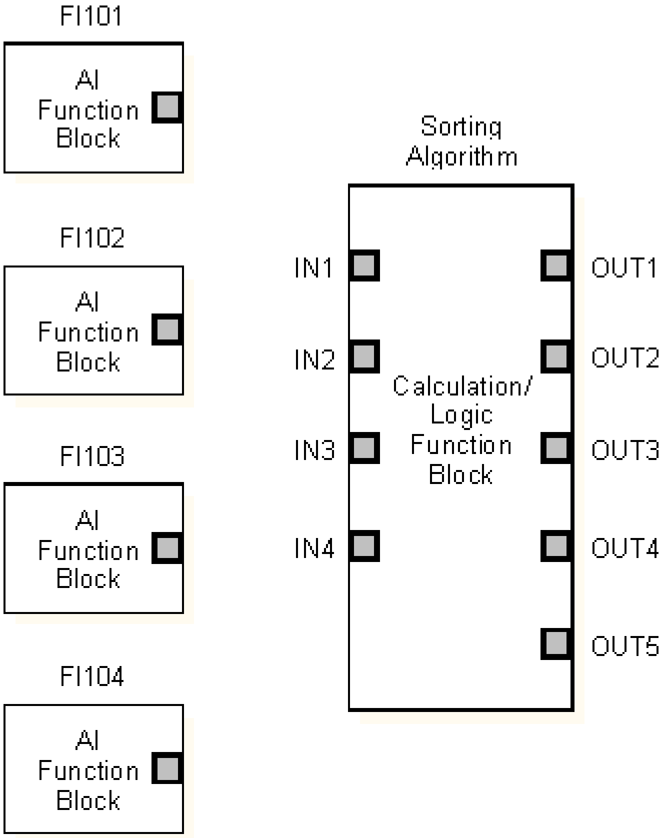
Calculation/Logic Function Block Diagram Example for Nonlinear Equation Correction

You configure the following expression in the Calculation/Logic block for this application:

OUT1 := 3 * expt(IN1, 3.0) + 2 * expt (IN1, 2.0) + IN1 + 1.0;

Application Example - Signal Ordering

Assume you want to arrange four input measurements in ascending order based on value and you want to write the sum of the four measurements to another function block. The following figure is the function block diagram for this application:



Module Block Logic For Signal Sorting

You could wire the Analog Input function blocks to the Calculation/Logic block and write the expression to utilize IN1 - IN4. However, this example illustrates the syntax for external references within a module. You configure the expressions as follows:

Note The expressions sort four external reference inputs from lowest to highest and writes the sorted values to the four outputs. The fifth output is set equal to the sum of the four external reference inputs.

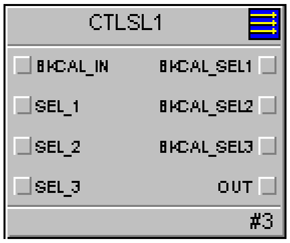
```
OUT1 :=  '/FI101.CV';
OUT2 :=  '/FI102.CV';
OUT3 :=  '/FI103.CV';
OUT4 :=  '/FI104.CV';
OUT5 := '/FI101.CV' + '/FI102.CV' + '/FI103.CV' + '/FI104.CV';
swap := FALSE;
REM First pass swap check
IF (OUT1 > OUT2) THEN
    temp := OUT1;
    OUT1 := OUT2;
    OUT2 := temp;
    swap := TRUE;
END_IF;
IF (OUT2 > OUT3) THEN
    temp := OUT2;
    OUT2 := OUT3;
    OUT3 := temp;
    swap := TRUE;
END_IF;
IF (OUT3 > OUT4) THEN
    temp := OUT3;
    OUT3 := OUT4;
    OUT4 := temp;
    swap := TRUE;
END_IF;
REM Second pass swap check
IF swap then
    swap := FALSE;
    IF (OUT1 > OUT2) THEN
        temp := OUT1;
        OUT1 := OUT2;
        OUT2 := temp;
        swap := TRUE;
    END_IF;
    IF (OUT2 > OUT3) THEN
        temp := OUT2;
        OUT2 := OUT3;
        OUT3 := temp;
        swap := TRUE;
    END_IF;
    IF (OUT3 > OUT4) THEN
        temp := OUT3;
        OUT3 := OUT4;
        OUT4 := temp;
        swap := TRUE;
    END_IF;
END_IF;
REM Third pass swap check
IF swap then
    swap := FALSE;
    IF (OUT1 > OUT2) THEN
        temp := OUT1;
        OUT1 := OUT2;
        OUT2 := temp;
        swap := TRUE;
    END_IF;
    IF (OUT2 > OUT3) THEN
        temp := OUT2;
        OUT2 := OUT3;
        OUT3 := temp;
        swap := TRUE;
    END_IF;
    IF (OUT3 > OUT4) THEN
```

```
        temp := OUT3;
        OUT3 := OUT4;
        OUT4 := temp;
        swap := TRUE;
    END_IF;
END_IF;
REM Fourth pass swap check
IF swap then
    swap := FALSE;
    IF (OUT1 > OUT2) THEN
        temp := OUT1;
        OUT1 := OUT2;
        OUT2 := temp;
        swap := TRUE;
    END_IF;
    IF (OUT2 > OUT3) THEN
        temp := OUT2;
        OUT2 := OUT3;
        OUT3 := temp;
        swap := TRUE;
    END_IF;
    IF (OUT3 > OUT4) THEN
        temp := OUT3;
        OUT3 := OUT4;
        OUT4 := temp;
        swap := TRUE;
    END_IF;
END_IF;
```

Control Selector Function Block

The Control Selector (CTLSL) function block selects one of three control signals to perform override control to a PID function block. The block supports mode control. The outputs are calculated based on the actual operation mode, which is determined by parameter values and input value statuses.

There are no standard alarms in this function block. Custom alarms are supported.



Control Selector Function Block

BKCAL_IN is the analog input value and status from a downstream block's BKCAL_OUT output that is used for backward output tracking for bumpless transfer.

SEL_1 is the first input value to the selector.

SEL_2 is the second input value to the selector.

SEL_3 is the third input value to the selector.

BKCAL_SEL1 is the selector output value associated with SEL_1 for backward output tracking to an upstream PID function block.

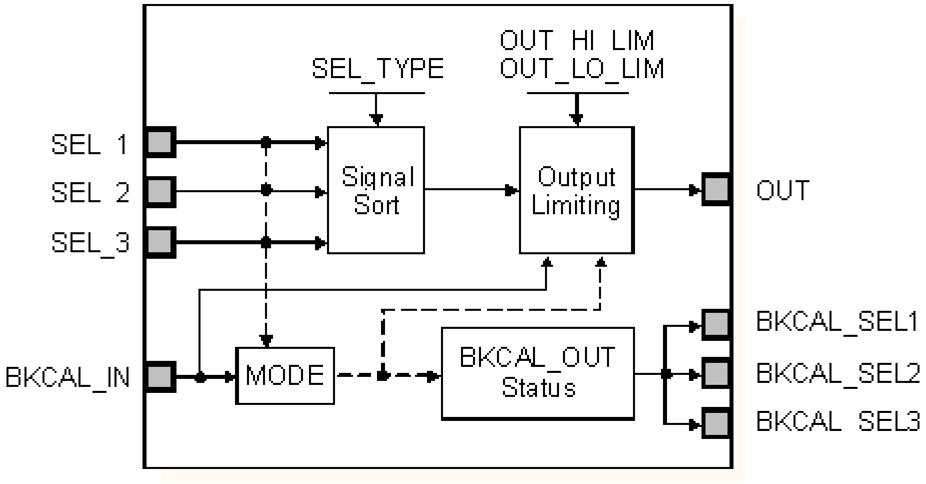
BKCAL_SEL2 is the selector output value associated with SEL_2 for backward output tracking to an upstream PID function block.

BKCAL_SEL3 is the selector output value associated with SEL_3 for backward output tracking to an upstream PID function block.

OUT is the output value and status.

Schematic Diagram - Control Selector Function Block

The following diagram shows the internal components of the Control Selector function block:



Control Selector Function Block Schematic Diagram

Block Execution - Control Selector Function Block

The Control Selector function block picks the high, low, or middle control signal from two or three PID function block primary outputs and places it at the Control Selector block's primary output. Three back calculation outputs are sent to upstream PID function blocks.

At the beginning of each scan period, the block calculates the actual mode according to the input status, target mode, and configured parameter settings. The block steps into the calculated actual mode or stays in the original mode.

Next, the block calculates the forward path primary output:

- In Initialization Manual (IMan) mode, the primary output is unchanged. The BKCAL_SEL1, BKCAL_SEL2, and BKCAL_SEL3 parameter values are passed to the upstream function blocks.
- In Manual (Man) mode, OUT can be set manually.
- In Automatic (Auto) mode, the block selects SEL_1, SEL_2, or SEL_3 as the primary output based on the selection type parameter (SEL_TYPE). The output value is sent back to all the upstream blocks.

Note In Auto mode, when any of the connected SEL_ 1, SEL_2 or SEL_3 inputs have Bad status during block execution, the block transitions its actual mode to Man. When the Bad SEL_1, SEL_2 or SEL_3 input transitions back to Good status, the block resumes Auto operation.

All the output parameter status values are set to indicate the status of the block and the status of the corresponding parameter.

Output Limit Scaling

On download, the limits are set to the high and low end of the scale. They might be changed by subsequent overrides, which are part of the download.

On run time, the following restrictions apply:

- OUT_HI_LIM is restricted to OUT_SCALE_HI+.1*(OUT_SCALE_HI-OUT_SCALE_LO).
- OUT_LO_LIM is restricted to OUT_SCALE_LO-.1*(OUT_SCALE_HI-OUT_SCALE_LO).

If the new scale causes a limit to be outside of these rules, the DeltaV system forces the limit within the rules.

The OUT parameters are not changed as a result of changing the scale or limits. However, the algorithm might change OUT on the next pass.

Output Selection and Limiting

You can limit the output by configuring the OUT_HI_LIM and OUT_LO_LIM parameters.

Modes - Control Selector Function Block

The Control Selector function block supports four modes:

Out of Service (OOS)

Initialization Manual (IMan)

Manual (Man)

Automatic (Auto)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Control Selector Function Block

The statuses of OUT, BKCAL_SEL1, BKCAL_SEL2 and BKCAL_SEL3 are mode dependent.

Manual Mode

The status of OUT is set to Good: Cascade, Constant. The status of BKCAL_SEL1, BKCAL_SEL2 and BKCAL_SEL3 are set to Good: Cascade, Not Invited (NI). The value of BKCAL_SEL1, BKCAL_SEL2 and BKCAL_SEL3 are set to the value of OUT.

Automatic Mode

OUT status is set to Good: Cascade. The limit status of the selected input (SEL_1, SEL_2 or SEL_3) is also copied to OUT.

The status and value of the BKCAL_SEL1, BKCAL_SEL2 or BKCAL_SEL3 parameter corresponding to the selected input is set equal to the OUT status and value. When the OUT status is limited, its limit status is also copied to the associated BKCAL_SEL[x] parameter and its value is set equal to the selected input value. When the OUT status is not limited but the BKCAL_IN status from the downstream block indicates it is limited, the BKCAL_IN limit status and value is copied to the associated BKCAL_SEL[x] parameter.

The status and value of the BKCAL_SEL[x] parameters that do not correspond to the selected input (the non-selected BKCAL_SEL[x] parameters) are set equal to the status and value of the BKCAL_SEL[x] parameter that corresponds to the selected input after its value and status is determined as above. The substatus of the non-selected BKCAL_SEL(x) parameters are set to Not Selected.

When the SEL_TYPE is Low value, the limit status of the non-selected BKCAL_SEL(x) parameters are set to High Limited. When the SEL_TYPE is High value, the limit status of the non-selected BKCAL_SEL(x) parameters are set to Low Limited. When the SEL_TYPE is Middle value, the limit status of the BKCAL_SEL(x) that corresponds to the lowest SEL(x) input is set to Low Limited. The limit status of the BKCAL_SEL(x) that corresponds to the highest SEL(x) input is set to High Limited.

Initialization Manual Mode

When the BKCAL_IN status indicates Initiate Request (IR), the status of OUT is set to Good: Cascade, Initiate Acknowledge (IA). Otherwise, the status of OUT is set to Good: Cascade. The value of OUT is set equal to the value of BKCAL_IN. The value and status of BKCAL_SEL1, BKCAL_SEL2 and BKCAL_SEL3 are set to the value and status of BKCAL_IN.

Out of Service Mode

The statuses of all output parameters are set to Bad: Out of Service.

Parameters - Control Selector Function Block

The following table lists the system parameters for the Control Selector function block:

Control Selector Function Block System Parameters

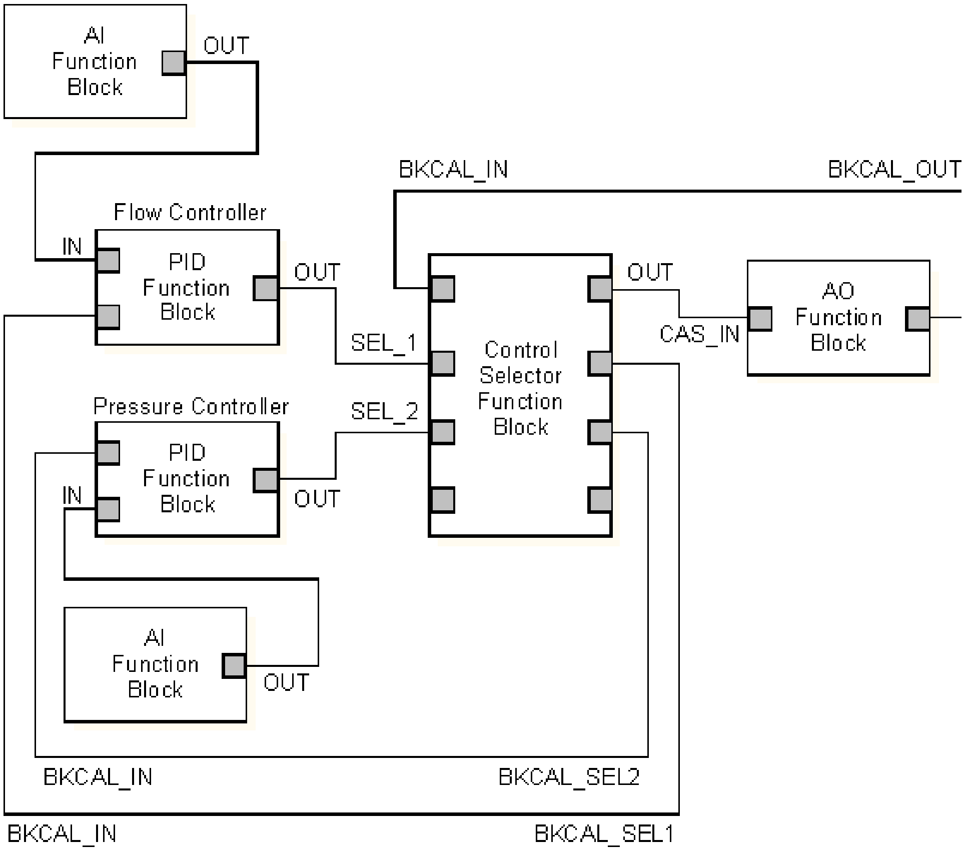
Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BKCAL_IN	Supplied by source	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BKCAL_SEL1	EU of OUT_SCALE	The selector output value associated with SEL_1 for backward output tracking.

Parameter	Units	Description
BKCAL_SEL2	EU of OUT_SCALE	The selector output value associated with SEL_2 for backward output tracking.
BKCAL_SEL3	EU of OUT_SCALE	The selector output value associated with SEL_3 for backward output tracking.
BLOCK_ERR	None	The summary of active error conditions associated with the block.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.
OUT	EU of OUT_SCALE	The analog output value and status.
OUT_HI_LIM	EU of OUT_SCALE	The maximum output value allowed.
OUT_LO_LIM	EU of OUT_SCALE	The minimum output value allowed
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
SEL_1	Supplied by source	The first input value to the selector.
SEL_2	Supplied by source	The second input value to the selector.
SEL_3	Supplied by source	The third input value to the selector.
SEL_TYPE	None	The selector type: High, Low, or Middle value.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

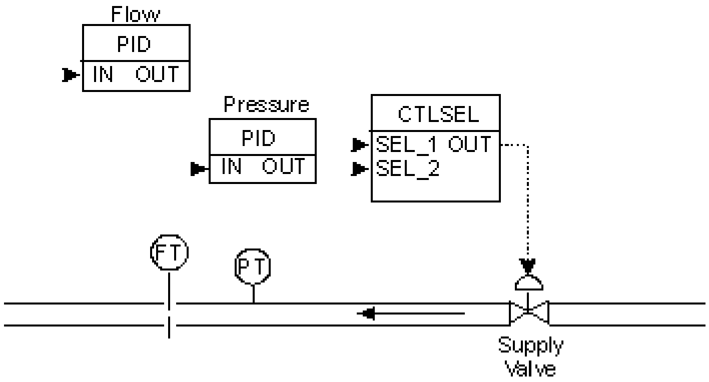
Application Information - Control Selector Function Block

The Control Selector function block is ideal for providing automatic override control. This function block can take three control signals as input. The user can select SEL_TYPE = Low, Medium, or High for various control configurations.



Control Selector Function Block Diagram Example

Use the Control Selector function block in a situation where flow is the primary control variable but pressure must be controlled in the event that it rises to a dangerous level. The following figure illustrates this example:



Flow Control with Pressure Override

In this example, both the flow and pressure PID blocks control a supply valve. The flow PID operates in a pressure regime below a safety limit. Since this is a low pressure for the pressure PID, it sends a high output signal to the supply valve, telling it to open more to increase the pressure.

At normal operating pressures, the pressure PID's high SP requests a maximum valve opening to increase pressure. The flow PID's signal is lower than that of the pressure PID. In this case, SEL_TYPE is set to Low so that the Control Selector function block sends the flow PID's signal on to the supply valve and blocks the pressure PID's signal.

If the flow stream's pressure rises to a dangerous level, the pressure PID sends a low signal to the supply valve, telling it to close. When this output signal falls below that from the flow PID, the Control Selector function block begins to pass the pressure PID's signal and block the flow PID's signal.

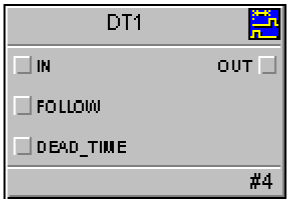
Deadtime Function Block

The Deadtime function block introduces a pure time delay in the value and status used in a signal path between two function blocks. The amount of delay introduced is determined by the DEAD_TIME parameter. You set DEAD_TIME manually or allow another function block that is calculating the desired delay to set the value automatically.

In Automatic (Auto) mode, the delayed input value and status are usually set to the block output (OUT). However, when the FOLLOW discrete input is active (FOLLOW = True), the block introduces no delay. A bias is added to the delayed input value to allow bumpless transfer for the transition from Manual (Man) to Auto mode. The bias is ramped to zero over the time specified by BAL_TIME.

The Deadtime function block supports signal status propagation and mode control. The actual mode calculation is based on the input status, target mode, and status handling options for the block.

There are no standard alarms in this function block. Custom alarms are supported.



Deadtime Function Block

IN is the analog input value and status.

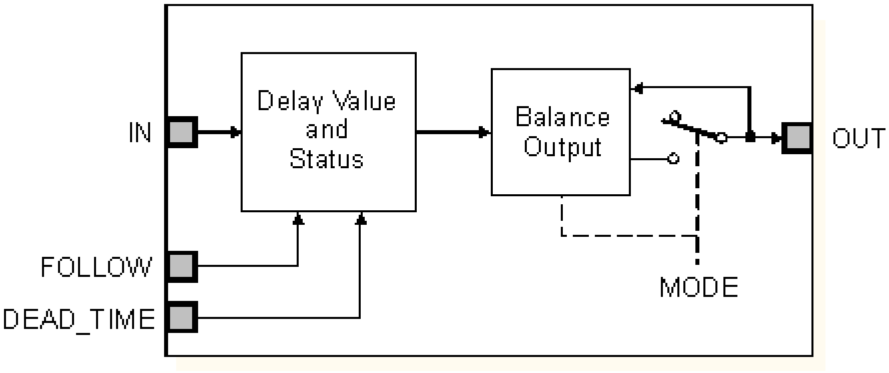
FOLLOW allows OUT to track IN.

DEAD_TIME is the delay time introduced in the input value, in seconds.

OUT is the analog output value and status.

Schematic Diagram - Deadtime Function Block

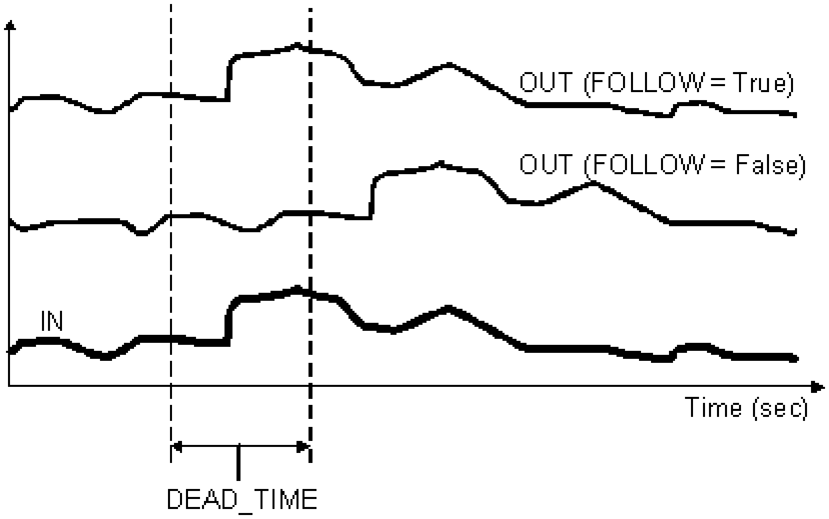
The following diagram shows the internal components of the Deadtime function block:



Deadtime Function Block Schematic Diagram

Block Execution - Deadtime Function Block

The following diagram shows the timed response of the Deadtime function block in Auto mode:



Deadtime Function Block Timing Diagram

You select the way the input signal is processed with parameter configuration and input wiring.

Delay Value and Status

The time delay to be introduced to the IN input is normally determined by the DEAD_TIME parameter. You enter the value of DEAD_TIME (in seconds), or another function block connected to IN can perform a calculation and enter the value automatically.

When the FOLLOW input parameter is active (True), no delay is introduced and the delayed input values stored by the block are initialized to the IN value and status.

Balance Output

In Man mode, the OUT value of the block can be entered manually. Thus, the delayed input value might not match the current block output. To avoid bumping the output on a mode change from Man to Auto, a bias value is added to the delayed signal so the resulting value matches the OUT value.

In Auto mode, the delayed input plus the bias is the value of the OUT parameter. The bias value added during the transition from Man to Auto mode is ramped from the OUT value calculation over the time defined by BAL_TIME.

Block Errors

The following condition is reported in the BLOCK_ERR parameter:

Out of Service – The block is in Out of Service (OOS) mode.

Modes - Deadtime Function Block

The Deadtime function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of the block might be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Deadtime Function Block

In Auto mode, the status of IN is passed to the OUT parameter after the delay time specified by the DEAD_TIME parameter. Any good input status is shown as Good. The status of IN has no impact on the block mode.

In Man mode, the OUT status is Good: Non-cascade, Constant.

Parameters - Deadtime Function Block

The following table lists the system parameters for the Deadtime function block:

Deadtime Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for the internal value of the delay input signal to match the OUT value.

Parameter	Units	Description
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Deadtime function block are Block configuration error and Out of Service.
DEAD_TIME	Seconds	Time delay introduced in the input signal to generate the output value.
FOLLOW	None	Allows OUT to track IN (True [1] = active, False [0] = inactive).
IN	Supplied by source	The input value and status to be delayed by the block.
MODE	None	Parameter used to request and show the source of the output used by the block.
OUT	EU of IN	The primary value and status calculated by the block in Auto mode. OUT can be set manually in Man mode.

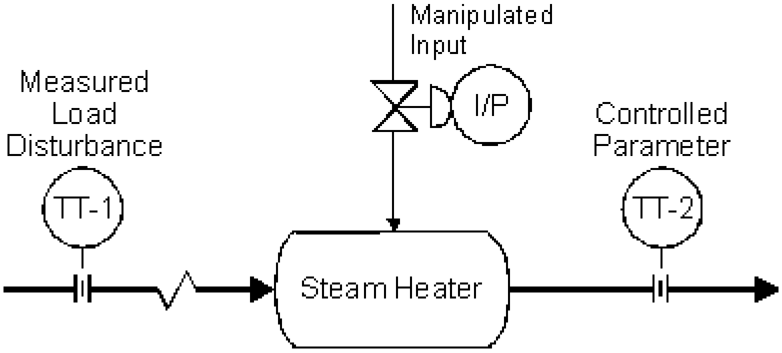
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Deadtime Function Block

One of the most common applications of the Deadtime function block is to provide dynamic compensation in a feedforward control strategy. In some processes, the time required to impact a controller parameter might be different for a measurable disturbance than for a manipulated process input. This is usually a result of transport delay in the process.

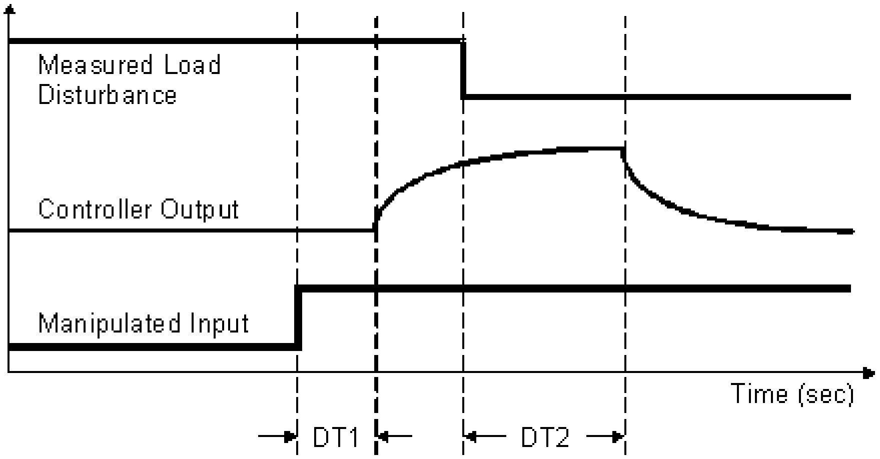
Application Example: Deadtime in a Heater Process

An typical heater process is shown in the following figure:



Deadtime Function Block Application Example for a Heater Process

The following diagram shows the difference in deadtime between the manipulated input and the measured load disturbance for this process:

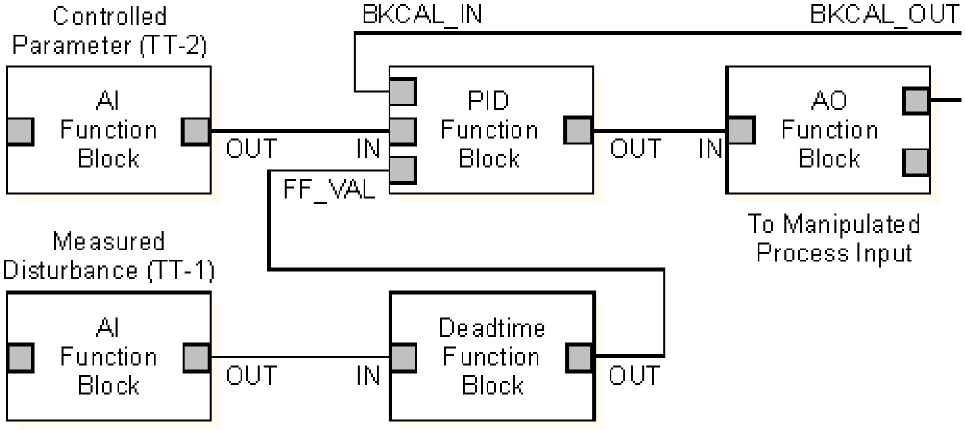


Deadtime Function Block Timing Diagram Example for a Heater Process

In this example, the Deadtime function block can be used to compensate for the differences in deadtime. When the flow rate to the heater is constant, the DEAD_TIME parameter should be set to a value of:

DT2 – DT1

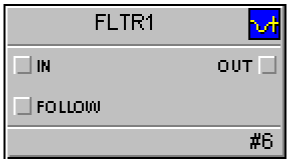
The following figure is an example function block diagram for this application:



Deadtime Function Block Diagram Example

Filter Function Block

The Filter (FLTR) function block applies an equation to filter changes in the input signal and generates a smooth output signal. The block supports signal status propagation. There are no modes or alarm detection in the Filter function block.



Filter Function Block

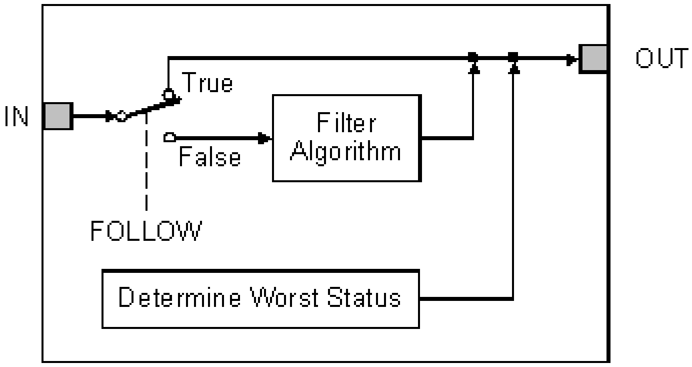
IN is the analog input value and status.

FOLLOW selects whether to filter the input or track the input.

OUT is the analog output value and status.

Schematic Diagram - Filter Function Block

The following diagram shows the internal components of the Filter function block:



Filter Function Block Schematic Diagram

Block Execution - Filter Function Block

The Filter function block executes a first-order lag algorithm (a low-pass filter) to filter out high frequency components of the input signal (spikes) and to pass through the low frequency components. The transfer functions for the Filter function block are as follows:

If TIMECONST <> 0:

$$OUT = x + (IN - x) * \left(1 - e{*}{*}^{-}\left[\frac{\Delta t}{TIMECONST}\right]\right)$$

If TIMECONST = 0:

$$OUT = x + \left(IN - x\right) * \left[1 - e{*}{*}\left[{}^{-}\left(\infty * \Delta t\right)\right]\right]$$

where:

x = the value of OUT at the cycle when IN last changed

t = the elapsed time since the cycle when IN last changed, in seconds

When the FOLLOW parameter is set True (1), the filter algorithm is disabled and the block output (OUT) equals the input signal (IN).

Status Handling - Filter Function Block

The output status is set to the worse status on IN and FOLLOW.

This is determined by first checking the quality (Bad=0, Uncertain=1, GoodProcess=2, and GoodCascade=3). If one status has a lower value than the other, then it is determined to be worse. If the quality value is the same, then the quality substatuses of the two are compared, and the higher substatus is determined to be worse. If the quality substatus is the same for both, then the limit fields of the status are checked with the higher limit being determined as worse.

Parameters - Filter Function Block

The following table lists the system parameters for the Filter function block.

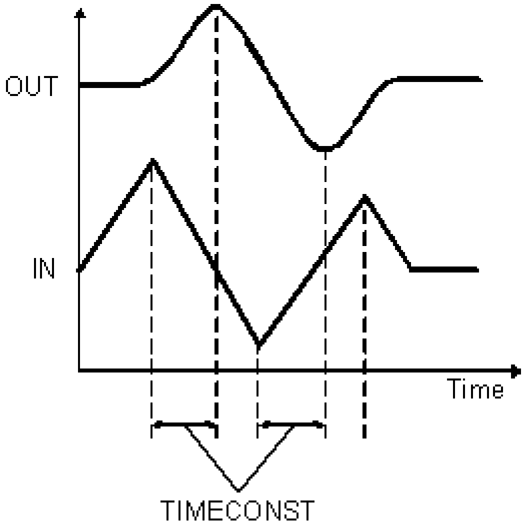
Filter Function Block System Parameters

Parameter	Units	Description
FOLLOW	None	Selects whether to filter the input (0) or track the input (1). When FOLLOW is True (1), the filter algorithm does not execute and OUT tracks IN.
IN	Determined by source	The analog input value and status.
OUT	Determined by IN	The analog output value and status.
TIMECONST	Seconds	The time constant. Specifies the time for OUT to equal 63% of IN.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Filter Function Block

The Filter function block smoothes out changes in an input signal. The following diagram shows the effect of the filter function on the block's output:



Filter Function Block Filtering Effect Example

You can use the Filter function block to filter a noisy analog input and keep the process stable. The amount of time for the output to equal the input is specified according to process considerations.

The TIMECONST parameter is used to vary the amount of time it takes for the output to equal the input. When the time required for the new input value to equal the output is 10 seconds:

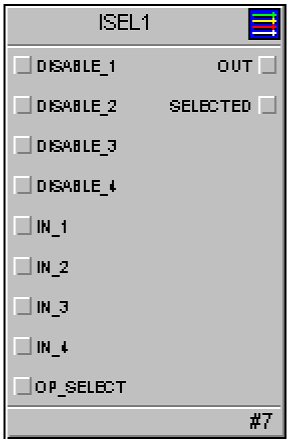
$$\text{TIMECONST} = \frac{10 \text{ seconds}}{5 \text{ time constants}} = 2$$

A time constant of 2 means that after 2 seconds, the output represents 63% of the new input value.

Input Selector Function Block

The Input Selector function block is a mathematical and logical input calculation block. The Input Selector chooses an output based on up to 4 inputs. This block supports mode control (Auto, Manual, Out of Service).

There is no standard alarm detection in the Input Selector function block.



Input Selector Function Block

Note The Input Selector function block is not designed for use in forward control paths.

DISABLE_1 when True, is used to disable IN_1. Disabled inputs are not used in by any of the select algorithms.

DISABLE_2 when True, is used to disable IN_2. Disabled inputs are not used in by any of the select algorithms.

DISABLE_3 when True, is used to disable IN_3. Disabled inputs are not used in by any of the select algorithms.

DISABLE_4 when True, is used to disable IN_4. Disabled inputs are not used in by any of the select algorithms.

IN_1 is the first input value.

IN_2 is the second input value.

IN_3 is the third input value.

IN_4 is the fourth input value.

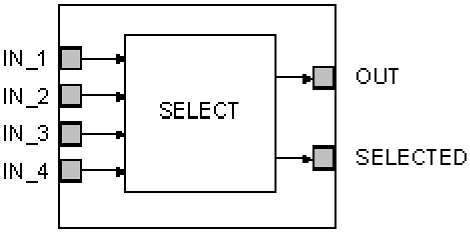
OP_SELECT if non-zero, selects the corresponding input (that is, 2 selects IN_2). If zero, the algorithm selected by SELECT_TYPE selects input.

OUT is the output value and status.

SELECTED indicates the selected input or the number of usable inputs.

Schematic Diagram - Input Selector Function Block

The following diagram shows the internal components of the Input Selector function block:



Input Selector Function Block Schematic Diagram

Block Execution - Input Selector Function Block

The Input Selector function block supports up to 4 inputs and selects the inputs either by direct selecting (OP_SELECT) or by applying a selected algorithm (SELECT_TYPE).

Direct Selecting Inputs

The parameter OP_SELECT is used to select a particular input by setting the value of OP_SELECT to the input number. For example, set OP_SELECT to 2 and IN_2 is direct selected. If OP_SELECT is zero, then the value of SELECT_TYPE is used.

Selection Types

The parameter SELECT_TYPE is configured for the selection algorithm for the block. The selection algorithms supported by Input Selector are as follows:

MAX Selects the highest valued input from those inputs that are not bad and are not disabled.

MIN Selects the lowest valued input from those inputs that are not bad and are not disabled.

MID Selects the mid valued input from those inputs that are not bad and are not disabled.

FIRST_GOOD Selects the first not disabled and not bad input, starting from IN_1.

AVG Averages those inputs that are not bad and are not disabled.

HOT_BACKUP Maintains the input selected from the previous scan, provided it is still usable. If it is not usable, HOT_BACKUP advances to the next usable input, counting up from the selected input number from the previous scan.

Modes - Input Selector Function Block

The Input Selector function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of a block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Input Selector Function Block

Uncertain if Man mode

Use Uncertain as Good

For complete descriptions of the supported status options, refer to the Status Options topic.

The output status depends on the following rules.

Status quality

- If the number of Uncertain or Good inputs is less than MIN_GOOD, OUT and SELECTED statuses are set to Bad.
- If an input is disabled and selected using OP_SELECT, the status of OUT and SELECTED will be the status of that input.

Substatus

- For SELECT_TYPE = MAX, MIN, FIRST_GOOD, and MID with an odd number of inputs, propagate substatus as is.
- For AVG or MID with even numbers of inputs, propagate NonSpecific.

Limit status

- If SELECT_TYPE = MID with an even number of inputs, set OUT limit status to NotLimited.

Parameters - Input Selector Function Block

The following table lists the parameters for the Input Selector function block.

Input Selector Function Block Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined BAD condition. The user selects a subset of block error (BLCOK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.

Parameter	Units	Description
BLOCK_ERR	None	The summary of active error conditions associated with the block. The possible block errors are Block configuration error, Simulate active, Local override, Input failure/process variable has Bad status, Output failure, Readback failed, Out of Service, and Other. Each function block reports none or a subset of these error conditions.
DISABLE_1	None	This parameter, when true, disables IN_1. A disabled input is not used by SELECT_TYPE configured algorithms.
DISABLE_2	None	This parameter, when true, disables IN_2. A disabled input is not used by SELECT_TYPE configured algorithms.
DISABLE_3	None	This parameter, when true, disables IN_3. A disabled input is not used by SELECT_TYPE configured algorithms.
DISABLE_4	None	This parameter, when true, disables IN_4. A disabled input is not used by SELECT_TYPE configured algorithms.
IN_1	Determined by supplying block or source.	The first analog input value and status.
IN_2	Determined by supplying block or source.	The second analog input value and status.
IN_3	Determined by supplying block or source.	The third analog input value and status.
IN_4	Determined by supplying block or source.	The fourth analog input value and status.
MIN_GOOD	None	Minimum number of usable inputs for the result to be not bad.
MODE	None	The mode record of the block. MODE contains the actual, target, permitted, and normal modes. In some function blocks, this parameter is used to request and show the source of the setpoint, the source of the output, and/or the block operating state.
OP_SELECT	None	If non-zero, selects the corresponding input (that is, 2 selects IN_2). If zero, the algorithm selected by SELECT_TYPE selects input.
OUT	EU of OUT_SCALE or Percent or EU of IN	The analog output value and status. The number of outputs is an extensible parameter in some blocks.

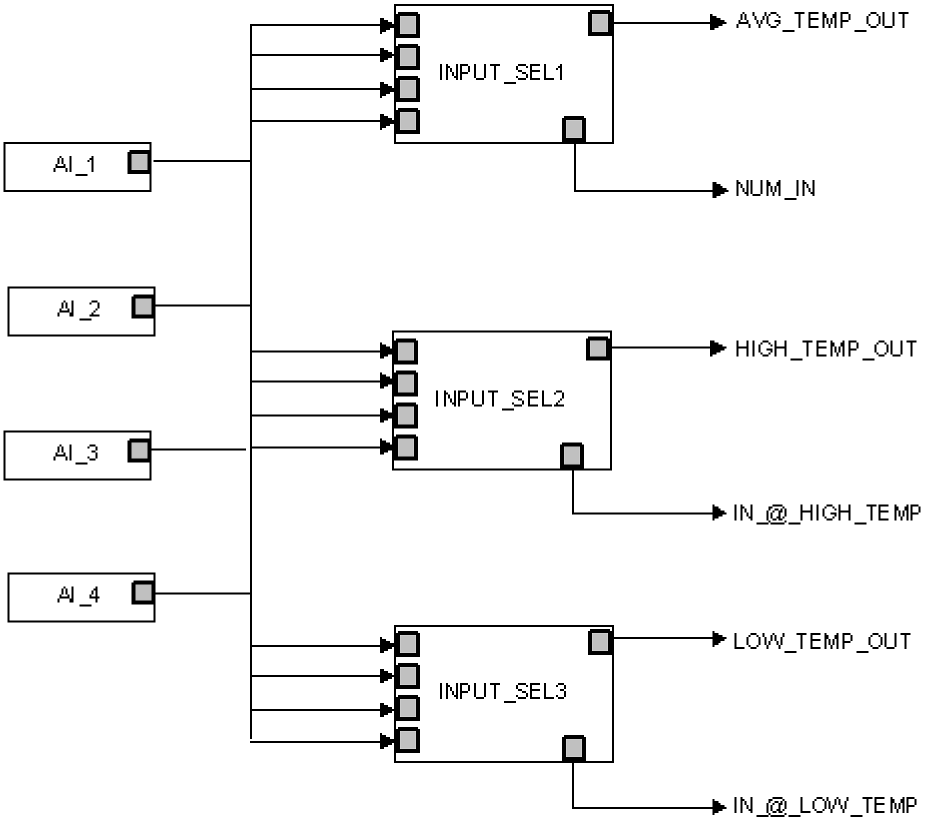
Parameter	Units	Description
SELECT_TYPE	None	If OP_SELECT is zero, determines the selection type. The following are valid selection types:
First Good	Minimum	Maximum
Middle	Average	Hot Backup
SELECTED	None	Output indicating the selected input or the number of usable inputs.
STATUS_OPTS	None	Status options. Allows you to select options for status handling and processing. The status options are Target to Manual if Bad IN and BAD if Limited. Each function block uses one or none of the status options.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.

Application Information - Input Selector Function Block

The Input Selector function block can be used to report average zone temperature with maximum and minimum temperature conditions on a group of temperature transmitters.

To use the Input Selector function block in this way, create a module of three (3) Input Selector function blocks and the analog inputs from the temperature transmitters (for this example, use four (4) analog inputs).

The first Input Selector function block continuously monitors average temperature and validates data only if three or more field inputs are present. The second and third Input Selector function blocks monitor the maximum and minimum temperatures in the zone and which indicators are responsible for reporting those temperatures.

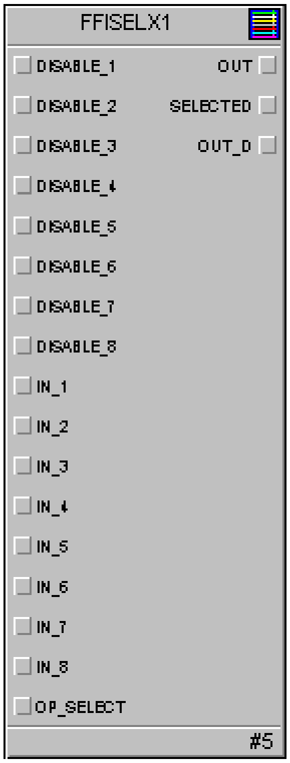


Input Selectors Used in Temperature Monitoring

This example provides a simple way of handling input data from a continuous process furnace for subsequent reporting and burner/blower management.

Input Selector Extended Function Block

The ISELX function block is a mathematical and logical input calculation block. The block chooses an output based on up to 8 inputs. This block supports mode control (Auto, Manual, Out of Service).



Input Selector Extended Function Block

Note The Input Selector Extended function block is not designed for use in forward control paths.

DISABLE_1 through DISABLE_8, when True, disable the corresponding IN_*n* input. For example, DISABLE_1 disables IN_1, DISABLE_2 disables IN_2, and so on. The SELECT_TYPE algorithms do not use disabled inputs.

IN_1 through IN_8 are the input values.

OP_SELECT, if non-zero, selects the corresponding input (for example, *2* selects IN_2). If zero, the algorithm selected by the SELECT_TYPE parameter selects input.

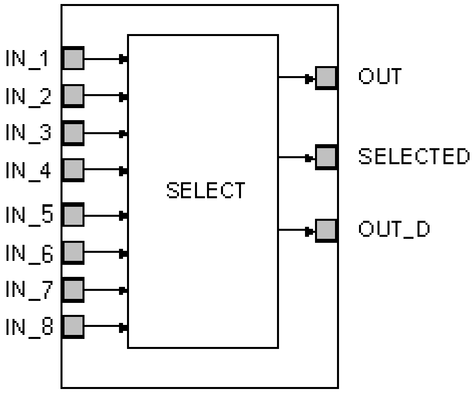
OUT is the analog output value and status.

SELECTED indicates the selected input or the number of usable inputs.

OUT_D is the discrete output value and status that signals a selected alarm condition.

Schematic Diagram - Input Selector Extended Function Block

The following diagram shows the internal components of the Input Selector Extended function block:



Input Selector Extended Function Block Schematic Diagram

Block Execution - Input Selector Extended Function Block

The Input Selector Extended function block supports up to 8 inputs and selects the inputs either by direct selecting (OP_SELECT) or by applying a selected algorithm (SELECT_TYPE).

Direct Selecting Inputs

The parameter OP_SELECT is used to select a particular input by setting the value of OP_SELECT to the input number. For example, set OP_SELECT to 2 and IN_2 is direct selected. If OP_SELECT is zero, then the value of SELECT_TYPE is used.

Selection Types

The parameter SELECT_TYPE is configured for the selection algorithm for the block. The selection algorithms supported by Input Selector Extended are as follows:

MAX Selects the highest valued input from those inputs that are not bad and are not disabled.

MIN Selects the lowest valued input from those inputs that are not bad and are not disabled.

MID Selects the mid valued input from those inputs that are not bad and are not disabled.

FIRST_GOOD Selects the first not disabled and not bad input, starting from IN_1.

AVG Averages those inputs that are not bad and are not disabled.

HOT_BACKUP Maintains the input selected from the previous scan, provided it is still usable. If it is not usable, HOT_BACKUP advances to the next usable input, counting up from the selected input number from the previous scan.

Block Errors

The following 16 conditions can be reported in the BLOCK_ERR parameter:

- **Other**
- **Block Configuration Error**: the selected channel carries a measurement that is incompatible with the engineering units selected in XD_SCALE, the L_TYPE parameter is not configured, or CHANNEL = zero.
- **Link Configuration Error**
- **Simulate Active**: Simulation is enabled and the block is using a simulated value in its execution.
- **Local Override**
- **Device Fault State Set**
- **Device Needs Maintenance Soon**
- **Input Failure/Process Variable has *Bad Status***: The hardware is bad, or a bad status is being simulated.
- **Output Failure**: The output is bad based primarily upon a bad input.
- **Memory Failure**
- **Lost Static Data**
- **Lost NV Data**
- **Readback Failed**
- **Device Needs Maintenance Now**
- **Power Up**
- **Out of Service**: The actual mode is Out of Service.

Modes - Input Selector Extended Function Block

The Input Selector Extended function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of a block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Alarm Detection - Input Selector Extended Function Block

Process alarm detection is based on the OUT value and the following user-specified alarm limits:

- High (HI_LIM)
- High high (HI_HI_LIM)
- Low (LO_LIM)
- Low low (LO_LO_LIM)

You can configure the ALM_SEL parameter to select the above alarm conditions which, when detected, set the OUT_D parameter, which can be linked to other function blocks for use in alarm detection.

Status Handling - Input Selector Extended Function Block

Uncertain if Man mode

Use Uncertain as Good

For complete descriptions of the supported status options, refer to the Status Options topic.

The output status depends on the following rules:

Status quality

- If the number of Uncertain or Good inputs is less than MIN_GOOD, OUT and SELECTED statuses are set to Bad.
- If an input is disabled and selected via OP_SELECT, the status of OUT and SELECTED will be the status of that input.

Substatus

- For SELECT_TYPE = MAX, MIN, FIRST_GOOD, and MID with an odd number of inputs, simply propagate substatus as is.
- For AVG or MID with even numbers of inputs, propagate NonSpecific.

Limit status

- If SELECT_TYPE = MID with an even number of inputs, set OUT limit status to NotLimited.

Parameters - Input Selector Extended Function Block

The following table lists the system parameters for the Input Selector Extended function block.

Input Selector Extended Function Block Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ACK_OPTION	None	Enables or disables automatic acknowledgment of Fieldbus device alarms. The default value is 0 (disabled). Set the value of this parameter to 65535 to enable the parameter before downloading the ISELX block. 0 = disabled (the default) 65535 = enabled
ALARM_HYS	Percent	The amount the alarm value must return within the alarm limit before the associated active alarm condition clears. ALARM_HYS is limited to 50% of scale.

Parameter	Units	Description
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.
ALM_SEL	None	Used to select the process alarm conditions that cause the OUT_D parameter to be True (Active).
AVG_USE	None	The minimum number of parameters the averaging calculation uses. An equal number of high and low values are dropped, leaving a number of values equal to or one greater than the value specified in AVG_USE. Example 1: AVG_USE is 4 and the number of connected inputs is 6. The highest and lowest values are dropped before calculating the average. Example 2: AVG_USE is 2 and the number of connected inputs is 7. The two highest and two lowest values are dropped and the average is calculated from the middle three inputs.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined BAD condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block.
DISABLE_*n* where *n* = 1 through 8	None	This parameter, when true, disables the corresponding input IN_*n*. A disabled input is not used by SELECT_TYPE configured algorithms.
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.
HI_HI_LIM	EU of OUT_RANGE	The setting for the alarm limit used to detect the high high alarm condition.

Parameter	Units	Description
HI_HI_PRI	None	The priority of the alarm detection associated with HI_HI_LIM and activated by HI_HI_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).
HI_LIM	EU of OUT_RANGE	The setting for the alarm limit used to detect the high alarm condition.
HI_PRI	None	The priority of the alarm detection associated with HI_LIM and activated by HI_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).
IN_*n* where *n* = 1 through 8	Determined by supplying block or source.	The analog input value and status corresponding to input number *n*.
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.
LO_LIM	EU of OUT_RANGE	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.
LO_LO_LIM	EU of OUT_RANGE	The setting for the alarm limit used to detect the low low alarm condition.
LO_LO_PRI	None	The priority of the alarm detection associated with LO_LO_LIM and activated by LO_LO_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).
LO_PRI	None	The priority of the alarm detection associated with LO_LIM and activated by LO_ACT. The value must be between 3 and 15 with 3 being a logged event. The values of 4 through 15 are priorities in ascending order of importance (that is, the higher the number, the greater the priority).
MIN_GOOD	None	Minimum number of usable inputs for the result to be not bad.

Parameter	Units	Description
MODE	None	The mode record of the block. MODE contains the actual, target, permitted, and normal modes. In some function blocks, this parameter is used to request and show the source of the setpoint, the source of the output, and/or the block operating state.
OP_SELECT	None	If non-zero, selects the corresponding input (that is, 2 selects IN_2). If zero, the algorithm selected by SELECT_TYPE selects input.
OUT	EU of OUT_RANGE	The analog output value and status.
OUT_D	None	The discrete output value and status.
OUT_RANGE	None	The display scaling for the output (OUT). It has no effect on the block. You can set the high and low values, the engineering units, and the number of digits to the right of the decimal point.
SELECT_TYPE	None	If OP_SELECT is zero, determines the selection type. The following are valid selection types: First Good Minimum Maximum Middle Average Hot Backup
SELECTED	None	Output indicating the selected input or the number of usable inputs.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.
STATUS_OPTS	None	Status options. Allows you to select options for status handling and processing. The status options available for the ISELX block are: Uncertain if Man mode Use Uncertain as Good
STRATEGY	None	The strategy field can be used to help group blocks. This data is not checked or processed by the block.

Application Information - Input Selector Extended Function Block

Application Example

Use the ISLEX block for applications using a fieldbus device such as a Rosemount 848T temperature transmitter. Use the SELECT_TYPE parameter to select an input or average the eight inputs and send the output to a PID block.

Lead/Lag Function Block

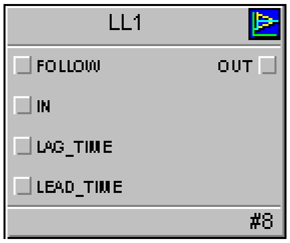
The Lead/Lag (LL) function block provides dynamic compensation for an input value. The block can apply a lead time function, a lag time function, or a combination of the two. A specified gain is applied to the compensated value and the value is high/low-limited based on the block mode.

The lead time and lag time values are specified by the LEAD_TIME and LAG_TIME parameters. You can specify a gain factor to be applied to the compensated value with the GAIN parameter.

The compensated value is high- and low-limited based on the block's mode. In Automatic (Auto) mode, the value of IN after dynamic compensation is multiplied by the GAIN value and written to OUT. However, when the FOLLOW discrete input is active (FOLLOW = True), no dynamic compensation is introduced by the block.

A bias value is added to the compensated input value to allow bumpless transfer on the transition from Manual (Man) or Out of Service (OOS) to Auto mode. The bias value is ramped to zero over the time specified by the balance time parameter (BAL_TIME).

The Lead/Lag function block supports signal status propagation. There are no standard alarms in this function block. Custom alarms are supported.



Lead/Lag Function Block

FOLLOW is the value that allows OUT to track IN * GAIN.

IN is the input value and status.

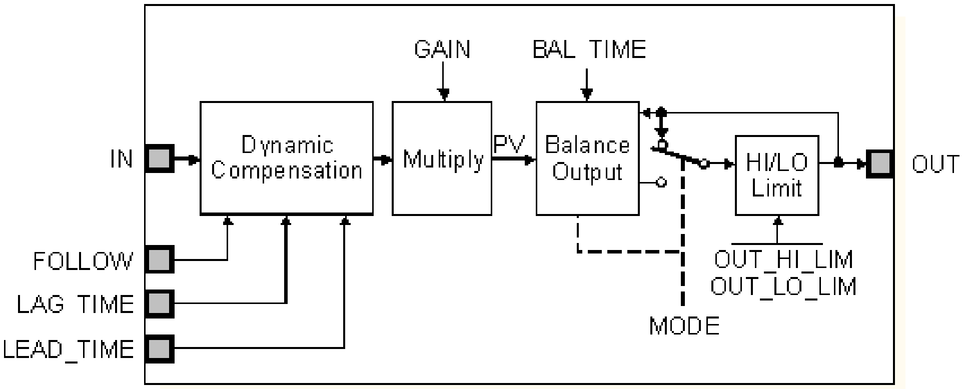
LAG_TIME specifies the lag time constant.

LEAD_TIME specifies the lead time constant.

OUT is the output value and status.

Schematic Diagram - Lead/Lag Function Block

The following diagram shows the internal components of the Lead/Lag function block:



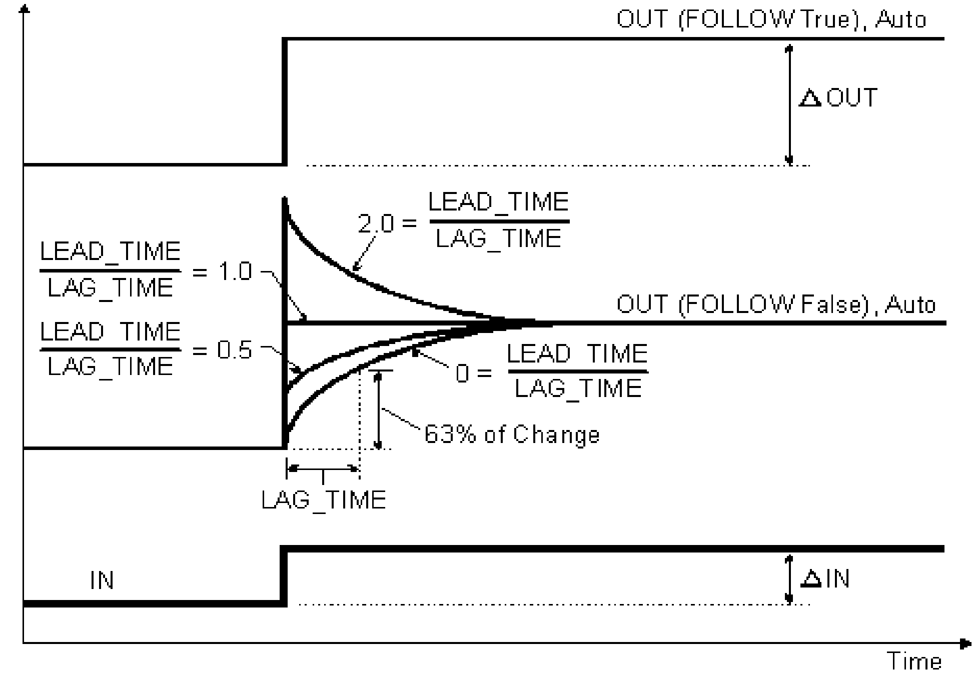
Lead/Lag Function Block Schematic Diagram

Block Execution - Lead/Lag Function Block

In the Lead/Lag function block,

$$GAIN = \frac{\Delta OUT}{\Delta IN}$$

The following diagram shows the timed response of the block.



Lead/Lag Function Block Timing Diagram

You select the way the input signal is processed with parameter configuration and input wiring.

Dynamic Compensation

The dynamic compensation that is applied to the IN input is normally determined by the LAG_TIME and LEAD_TIME parameters. You can enter the value of these parameters manually, or another block can provide the value. The compensated input multiplied by the GAIN parameter is the block process variable (PV).

In Auto mode, the PV is normally reflected in the OUT parameter. However, when the PV value exceeds the output limits defined by OUT_HI_LIM and OUT_LOW_LIM, the limit value is used.

When the FOLLOW parameter is active (True), no dynamic compensation is applied and the result of IN multiplied by GAIN is reflected in the OUT parameter.

Balance Output

In Man mode, the OUT value can be entered manually. Thus, the dynamically compensated input value shown in PV might not match the current block output. To avoid bumping the output on a mode change from Man or OOS to Auto, a bias value is added to the compensated signal so the resulting value matches OUT.

In Auto mode, the compensated input plus the bias value is the OUT value. The bias value added during the transition from Man or OOS to Auto mode is ramped down to zero over the time defined by the BAL_TIME parameter.

Modes - Lead/Lag Function Block

The Lead/Lag function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The status of IN has no impact on the block's mode. The target mode of the block can be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Lead/Lag Function Block

In Auto mode, the status of IN is passed to OUT. Any good input status is shown as Good: Non-cascade. The OUT limit status indicates whether the OUT value has been limited.

In Man mode, the OUT status is always Good: Non-cascade, Constant.

Parameters - Lead/Lag Function Block

The following table lists the system parameters for the Lead/Lag function block:

Lead/Lag Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for PV to match the OUT value after a transition from Man or OOS to Auto mode.
BLOCK_ERR	None	The error condition associated with the block that indicates whether or not the block is Out of Service.
FOLLOW	None	Allows OUT to track IN * GAIN (True [1] = active, False [0] = inactive).
GAIN	None	The gain value applied to IN.
IN	Determined by source	The analog input value and status to be dynamically compensated by the block.
LAG_TIME	Seconds	The lag time constant applied in dynamic compensation.
LEAD_TIME	Seconds	The lead time constant applied in dynamic compensation.

Parameter	Units	Description
MODE	None	Parameter used to request and show the source of the output used by the block.
OUT	EU of IN	The primary value and status calculated by the block in Auto mode. OUT can be set manually in Man mode.
OUT_HI_LIM	EU of IN	The maximum output value allowed.
OUT_LO_LIM	EU of IN	The minimum output value allowed.
PV	EU of IN	The process variable used in block execution and alarm limit detection.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Lead/Lag Function Block

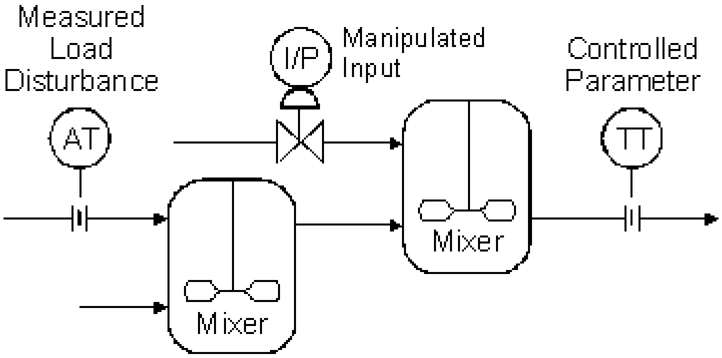
The Lead/Lag function block is useful for a variety of process control applications. Two typical applications are the filtering of a measured or calculated value and feedforward dynamic compensation.

Application Example: Filtering a Measured or Calculated Value

You use the Lead/Lag function block as a first-order filter by setting the lead time constant (LEAD_TIME) to zero. In this case, the amount of filtering applied is determined by the lag time constant (LAG_TIME).

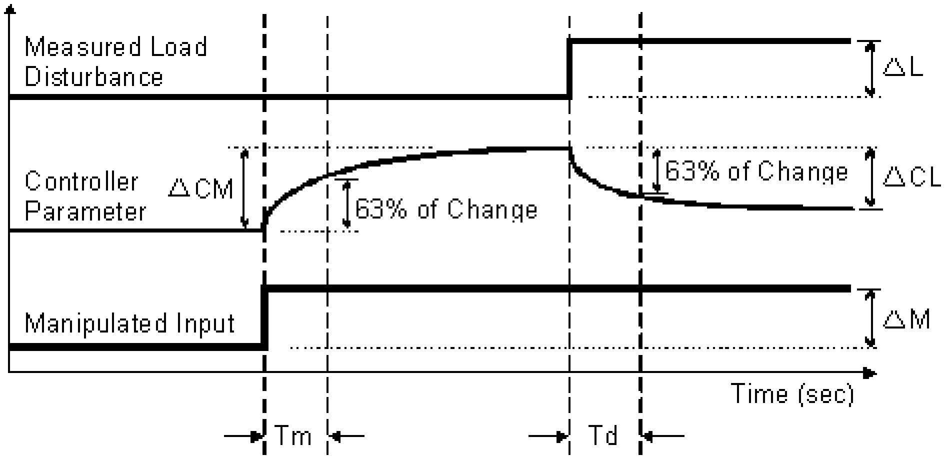
Application Example: Feedforward Dynamic Compensation

In some processes, the time required for the measurable disturbance and the manipulated process input to impact a controller parameter might be the same, but the associated response time (lag) might be different. The following figure shows an example of a mixing process that experiences this difference in lag times:



Lead/Lag Function Block Application Example for a Mixing Process

The following diagram shows the difference in lag time between the manipulated input and the measured load disturbance for this process:



Lead/Lag Function Block Timing Diagram Example for a Mixing Process

In this example, the Lead/Lag function block can be used to compensate the feedforward control action for the controlled parameter's differences in gain and dynamic response to changes in the load disturbance and the manipulated input. To do this, you include a Lead/Lag function block in the feedforward path. You set the LEAD_TIME parameter to Tm and the LAG_TIME parameter to Td.

If

$$KL = \left(\frac{\Delta CL}{\Delta L}\right)$$

where:

KL = process gain for load disturbance input

CL = steady state change in the controlled parameter for a step change in the load disturbance input (% of range)

L = step change in the load disturbance input (% of range)

and

$$KM = \frac{\Delta CM}{\Delta M}$$

where:

KM = process gain for manipulated input

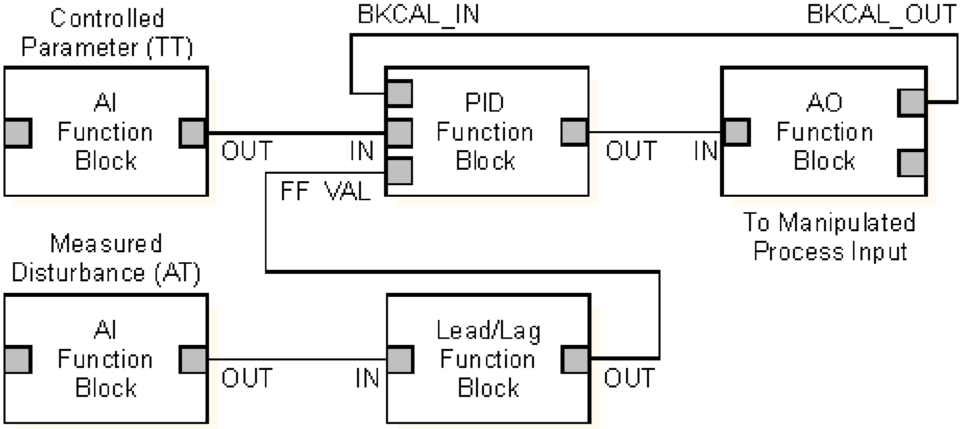
CM = steady state change in the controlled parameter for a step change in the manipulated input (% of range)

M = step change in the manipulated input (% of range),

you set the GAIN parameter to:

$$-\left(\frac{KL}{KM}\right)$$

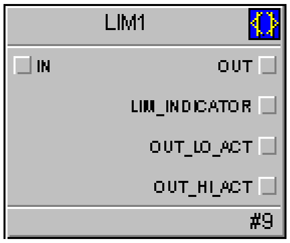
When the process response also includes significant delay, you can use a Deadtime function block to compensate for the delay. The following figure is an example of the function block diagram for this application:



Lead/Lag Function Block Diagram Example

Limit Function Block

The Limit (LIM) function block limits an input value between two reference values. The block supports signal status propagation. There are no modes or alarm detection in the Limit function block.



Limit Function Block

IN is the analog input value and status.

OUT is the analog output value and status.

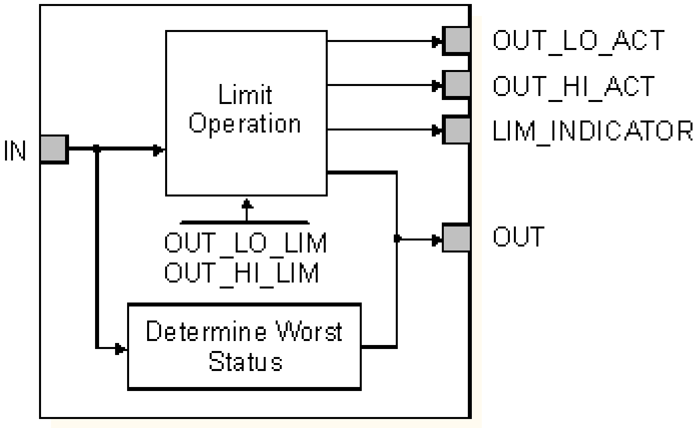
LIM_INDICATOR is set True (1) when the input is limited to the maximum value. It remains True until the input is limited to the minimum value, at which time it is set False (0).

OUT_LO_ACT is a Boolean value set True when the input is limited to the minimum value.

OUT_HI_ACT is a Boolean value set True when the input is limited to the maximum value.

Schematic Diagram - Limit Function Block

The following figure shows the internal components of the Limit function block:



Limit Function Block Schematic Diagram

Block Execution - Limit Function Block

The Limit function block restricts the output value between a high limit and a low limit. When IN is less than or equal to the configured minimum value (OUT_LO_LIM), OUT equals OUT_LO_LIM and OUT_LO_ACT is set True.

When IN is greater than or equal to the configured maximum value (OUT_HI_LIM), OUT equals OUT_HI_LIM and OUT_HI_ACT is set True.

OUT_HI_ACT and OUT_LO_ACT are set False.

When IN becomes greater than or equal to OUT_HI_LIM, LIM_INDICATOR is set True.

When IN becomes less than or equal to OUT_LO_LIM, LIM_INDICATOR is set False.

The following table shows an example of the Limit function block outputs when OUT_LO_LIM = 5 and OUT_HI_LIM = 90:

Limit Function Block Execution Example

IN	OUT	OUT_LO_ACT	OUT_HI_ACT	LIM_INDICATOR
0	5	True	False	False
5	5	False	False	Equal to the previous value
50	50	False	False	Equal to the previous value
90	90	False	False	Equal to the previous value
100	90	False	True	True

Status Handling - Limit Function Block

The statuses of the outputs (OUT, OUT_HI_ACT, and OUT_LO_ACT) are set to the input status. The status of LIM_INDICATOR is always Good.

Parameters - Limit Function Block

The following table lists the system parameters for the Limit function block:

Limit Function Block System Parameters

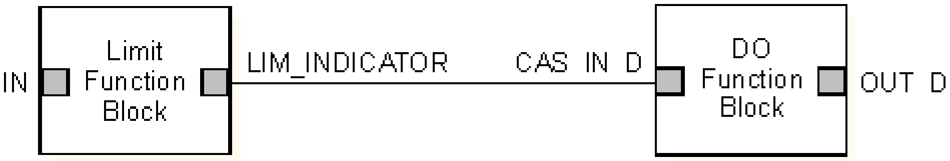
Parameter	Units	Description
IN	Determined by source	The analog input value and status.
LIM_INDICATOR	None	Discrete output value set True (1) when IN is limited to OUT_HI_LIM. It remains True until IN becomes limited by OUT_LO_LIM.
OUT	Determined by IN	The analog output value and status.
OUT_HI_ACT	None	Boolean value set True when the input is limited to the maximum value.

Parameter	Units	Description
OUT_HI_LIM	Determined by IN	The maximum output value allowed.
OUT_LO_ACT	None	Boolean value set True when the input is limited to the minimum value.
OUT_LO_LIM	Determined by IN	The minimum output value allowed.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Limit Function Block

You can use the Limit function block to start a pump that drains a tank when it is full. In this example, the input to the Limit block is the tank level. When LIM_INDICATOR becomes True, the block sends a signal to a Discrete Output (DO) function block to turn the pump on and drain the tank. When LIM_INDICATOR goes to False (when OUT_LO_LIM is reached) the pump shuts off. The following figure is the function block diagram for this example:



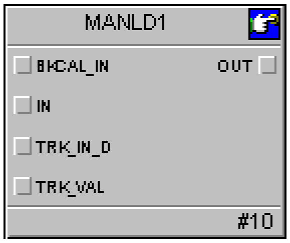
Limit Function Block Diagram Example

Manual Loader Function Block

The Manual Loader (MANLD) function block allows the block output to be set by an operator. The block supports output tracking and alarm detection.

You can force the output value to the tracking value by using the tracking inputs. This is useful for bumpless transfer. The BKCAL_IN input parameter sets the block output automatically for bumpless transfer when the downstream path is not complete.

You can connect other function blocks that require the manually entered value to the Manual Loader block output (OUT). In addition, you can connect an input measurement or calculation value that might be useful to an operator in making manual entries by connecting the associated block parameter to the Manual Loader block input (IN).



Manual Loader Function Block

BKCAL_IN is the value from another block's BKCAL_OUT used for backward output tracking for bumpless transfer.

IN is the analog input value and status.

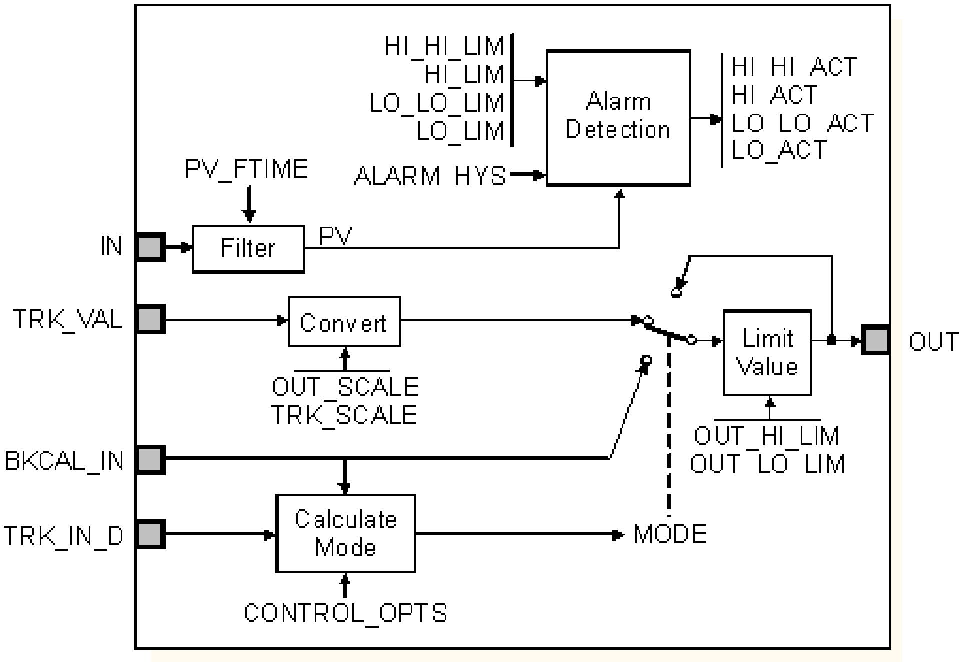
TRK_IN_D is the discrete input that initiates the external tracking function.

TRK_VAL is the value applied to OUT in Local Override (LO) mode.

OUT is the analog output value and status.

Schematic Diagram - Manual Loader Function Block

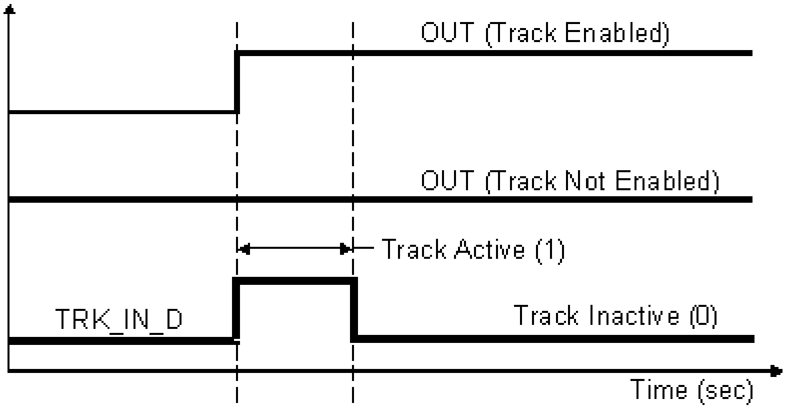
The following figure shows the internal components of the Manual Loader function block:



Manual Loader Function Block Schematic Diagram

Block Execution - Manual Loader Function Block

The following figure shows the timed response of the Manual Loader function block:



Manual Loader Function Block Timing Diagram

The Manual Loader function block allows an operator to control devices directly by setting the block output. You select the way the output is generated with parameter configuration and input wiring.

Input Filtering

You can filter the input value (IN) before it is stored in the PV parameter. You specify the filtering applied to the IN parameter with the filter time constant (PV_FTIME). The PV value is used in block alarm detection.

Track Input Conversion

The track input parameter (TRK_VAL) is converted to the range of the block output based on the track range parameter (TRK_SCALE) and the output range parameter (OUT_SCALE). In Local Override (LO) mode, the scaled value is used to set OUT.

Output Limit Scaling

On download, the limits are set to the high and low end of the scale. They might be changed by subsequent overrides, which are part of the download.

On run time, the following restrictions apply:

- OUT_HI_LIM is restricted to OUT_SCALE_HI+.1*(OUT_SCALE_HI-OUT_SCALE_LO).
- OUT_LO_LIM is restricted to OUT_SCALE_LO-.1*(OUT_SCALE_HI-OUT_SCALE_LO).

If the new scale causes a limit to be outside of these rules, the DeltaV system forces the limit within the rules.

The OUT parameters are not changed as a result of changing the scale or limits. However, the algorithm might change OUT on the next pass.

Output Selection and Limiting

In Manual (Man) mode, the value of OUT is determined by the operator. In LO mode, the converted track input is used as the output value. In Initialization Manual (IMan) mode, the path to the output is not complete and OUT is set to BKCAL_IN.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Local override – The block is in Local Override (LO) mode.

Out of Service – The block is in Out of Service (OOS) mode.

For complete descriptions of the supported control options, refer to the Control Options topic.

Modes - Manual Loader Function Block

The Manual Loader function block supports four modes:

Out of Service (OOS)

Initialization Manual (IMan)

Local Override (LO)

- Track Enable
- Track in Manual

Manual (Man)

The target mode can be restricted to one or more of the modes that are set by an operator (Man mode and OOS mode). During operation, you can only set the Man and OOS modes directly.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Note You can set control options in Manual or Out of Service mode only.

Refer to the Control Options topic for more information on control options.

Alarm Detection - Manual Loader Function Block

Block alarm detection is based on the PV parameter value. You can configure the alarm limits of the following standard alarms:

High (HI_LIM)
High high (HI_HI_LIM)
Low (LO_LIM)
Low low (LO_LO_LIM)

The amount the PV alarm value must return within the alarm limits before the active alarm condition clears (alarm deadband) is determined by the ALARM_HYS parameter.

Status Handling - Manual Loader Function Block

The status of the IN parameter is reflected in the PV status but has no impact on OUT. The status of OUT is determined by the status of BKCAL_IN and the actual mode of the block.

Parameters - Manual Loader Function Block

The following table lists the system parameters for the Manual Loader function block:

Manual Loader Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALARM_HYS	Percent of PV_SCALE	The amount the alarm value must return within the alarm limit before the associated active alarm condition clears. ALARM_HYS is limited to 50% of scale.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BKCAL_IN	EU of OUT_SCALE	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Manual Loader function block are Local override and Out of Service.
CONTROL_OPTS	None	Control options. Allow you to specify control strategy options. The supported control options for the Manual Loader function block are No OUT Limits in Manual, Track Enable, and Track in Manual.
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.

Parameter	Units	Description
HI_HI_LIM	EU of PV	The setting for the alarm limit used to detect the high high alarm condition.
HI_LIM	EU of PV	The setting for the alarm limit used to detect the high alarm condition.
IN	EU of PV	The input value that is filtered to obtain PV.
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.
LO_LIM	EU of PV	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.
LO_LO_LIM	EU of PV	The setting for the alarm limit used to detect the low low alarm condition.
MODE	None	Parameter used to request and show the source of the output used by the block.
OUT	EU of OUT_SCALE	The primary value and status of the block. OUT can be set by an operator in Man mode.
OUT_HI_LIM	EU of OUT_SCALE	The maximum output value allowed.
OUT_LO_LIM	EU of OUT_SCALE	The minimum output value allowed.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
PV	EU of PV_SCALE	The process variable used in block execution and alarm limit detection.
PV_FTIME	Seconds	The time constant of the first-order PV filter. It is the time required for a 63% change in the IN value.
PV_SCALE	None	The high and low scale values, the engineering units code, and the number of digits to the right of the decimal point associated with PV.
TRK_IN_D	None	Discrete input that initiates output tracking (True [1] = active, False [0] = inactive). TRK_IN_D is acted on only when the Track Enable and/or Track in Manual control options are selected.
TRK_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the external tracking value (TRK_VAL).
TRK_VAL	EU of OUT_SCALE	The analog input used in the external tracking function.

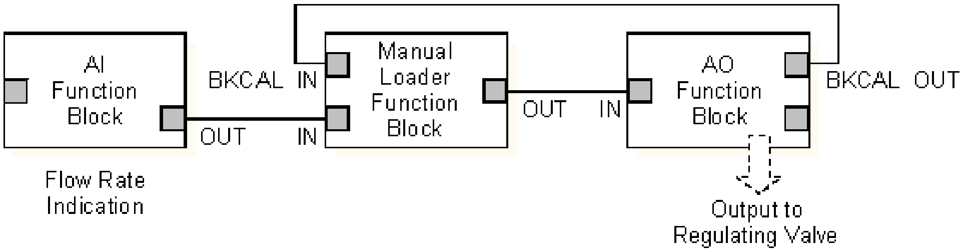
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Manual Loader Function Block

You use the Manual Loader function block in applications where an operator adjusts the process input manually (that is, no automatic control) and you want a measurement or calculated value related to this adjustment to be available for display or alarming.

Application Example: Manual Valve Adjustment

When the upstream and downstream operating conditions are constant, the flow through a regulating valve is relatively constant. If the flow rate is set by an operator, the valve can be positioned manually using an indication of the flow rate (such as flow rate, downstream pressure). Alarming on this indication can alert the operator that the operating condition has changed. The following figure shows the function block diagram for this example:



Manual Loader Function Block Diagram Example

PID Function Block

The PID function block combines all of the necessary logic to perform analog input channel processing, proportional-integral-derivative (PID) control with the option for nonlinear control (including error-squared and notched gain), and analog output channel processing.

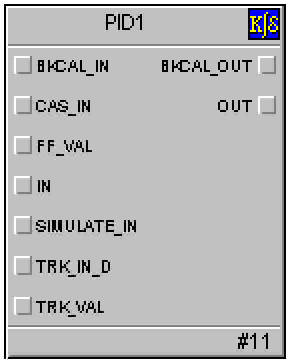
The PID function block supports mode control, signal scaling and limiting, feedforward control, override tracking, alarm limit detection, and signal status propagation. To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block through the SIMULATE_IN input.

In Cascade (Cas) mode, the setpoint (SP) is adjusted by a master controller. In Automatic (Auto) mode, the SP can be adjusted by the operator. In both Cas and Auto modes, the output is calculated with a standard or series PID equation form. In Manual (Man) mode, the block's output is set by the operator. The PID function block also has two remote modes, RCas and ROut. These modes are similar to Cas and Man modes except that SP and OUT are supplied by a remote supervisory program.

The PID function block can be connected directly to process I/O (in Delta V, but not in Fieldbus devices). It can also be connected to other function blocks through its IN and OUT parameters for cascade and other more complex control strategies.

You connect BKCAL_OUT to an upstream block's BKCAL_IN to prevent reset windup and provide bumpless transfer to closed loop control.

You can connect the tracking input (TRK_VAL) for externally controlled output tracking.



PID Function Block

BKCAL_IN is the analog input value and status from a downstream block's BKCAL_OUT output that is used by a block for bumpless transfer. This connection is necessary if the PID is a master to another controller in a cascade. Without the connection the slave controller will not make the transition to CAS and the master PID will never be active.

CAS_IN is the remote SP value from another block.

FF_VAL is the feedforward control input value and status.

IN is the connection for the process variable (PV) from another function block.

SIMULATE_IN is the input value and status used by the block instead of the analog measurement when simulation is enabled.

TRK_IN_D initiates the external tracking function.

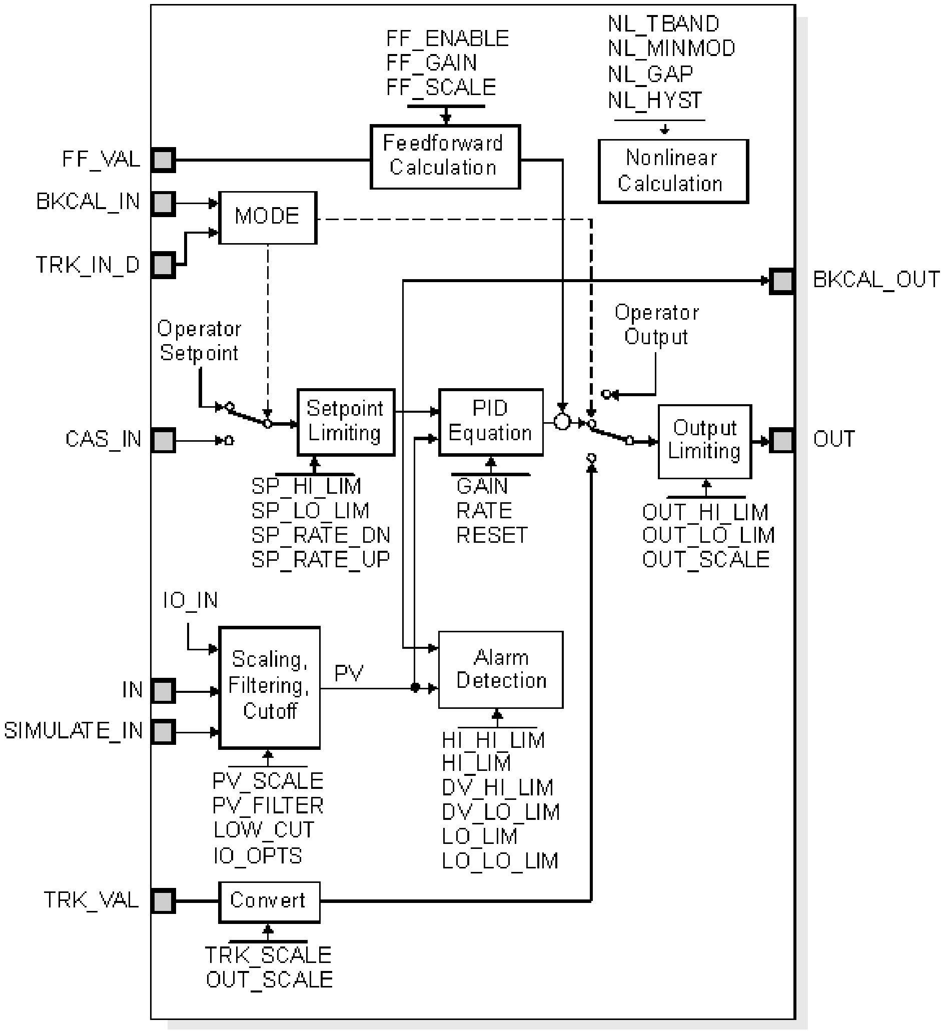
TRK_VAL is the value after scaling applied to OUT.

BKCAL_OUT is the value and status sent to an upstream block to prevent reset windup and provide bumpless transfer to closed loop control.

OUT is the block output value and status.

Schematic Diagram - PID Function Block

The following diagram shows the internal components of the PID function block:



PID Function Block Schematic Diagram

Block Execution - PID Function Block

The PID function block provides proportional (P) + integral (I) + derivative (D) control. Two PID equation forms are supported in the block, both forms supporting external reset and feedforward:

The standard form is a discrete implementation of:

$$OUT(s) = \pm GAIN_a \bullet \left(KNL \bullet \left(\frac{P(s) \bullet T_r s}{(T_r s + 1)} + \frac{E(s)}{(T_r s + 1)} \right) + \frac{D(s) \bullet T_r s \bullet T_d s}{(T_r s + 1)(\alpha T_d s + 1)} \right) + \frac{L(s) - F(s)}{(T_r s + 1)} + F(s)$$

The series form is a discrete implementation of:

$$OUT(s) = \pm GAIN_a \bullet \left(\frac{P(s) \bullet T_r s}{(T_r s + 1)} + \frac{E(s)}{(T_r s + 1)} + \frac{D(s) \bullet T_d s}{(\alpha T_d s + 1)} \right) + \frac{L(s) - F(s)}{(T_r s + 1)} + F(s)$$

For L = OUT (which is the same as OUT being unconstrained) and P = D = E the equations reduce to:

A conventional Standard PID with feedforward,

$$OUT(s) = GAIN_a \bullet \left(1 + \frac{1}{T_r s} + \frac{T_d s}{(\alpha T_d s + 1)} \right) \bullet E(s) + F(s)$$

and Series PID with derivative filter applied only to derivative action, with feedforward

$$OUT(s) = GAIN_a \bullet \left(1 + \frac{T_d s}{(\alpha T_d s + 1)} \right) \left(\frac{T_r s + 1}{T_r s} \right) \bullet E(s) + F(s)$$

Where:E(s) is error (SP-PV)

± is + for reverse acting and – for direct acting (Direct_Acting in CONTROL_OPTS)

KNL is nonlinear gain applied to P + I terms but not to D term. Nonlinear action is activated in FRSIPID_OPTS by selecting Use_Nonlinear_Gain_Modification.

P(s) is the variable to which proportional action is applied. P(s) is determined by parameters STRUCTURE and BETA (which sets the weighting factor for proportional action applied to SP change).

D(s) is the variable to which derivative action is applied. D(s) is determined by parameters STRUCTURE and GAMMA (which sets the weighting factor derivative action on SP change).

L(s) is the external reset input which is either from BKCAL_IN or OUT.

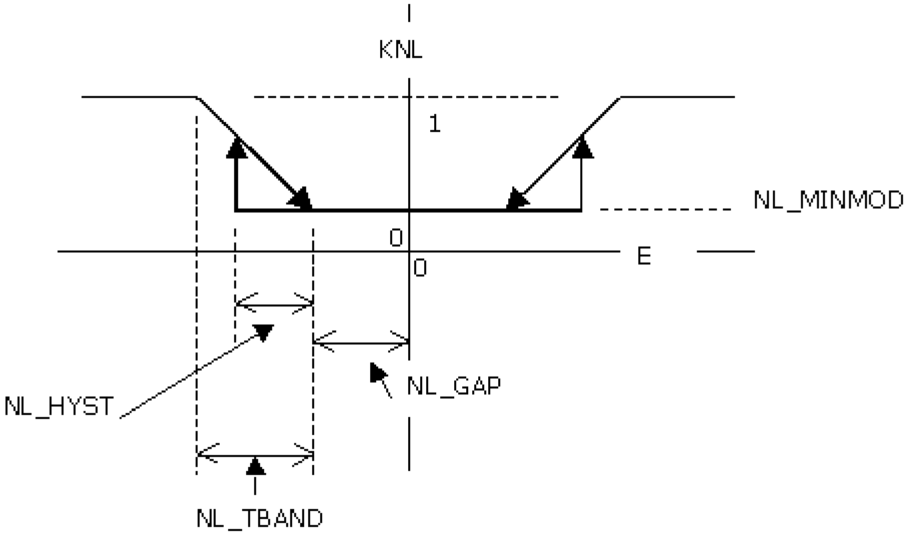
T_r is reset time (parameter RESET) in seconds.

T_d is derivative time (parameter RATE) in seconds

GAINa is normalized gain after scaling the parameter GAIN from PV to OUT (Delta V works in engineering units so it is necessary that the parameter GAIN be scaled to maintain the meaning of the normalized entry).

F(s) is the feedforward contribution.

The following diagram illustrates how the nonlinear tuning parameters are used in the calculation of KNL.



KNL Calculation

where:

NL_MINMOD is the gain applied when the absolute value of error is less than NL_GAP. To get deadband behavior, set NL_MINMOD to 0.

NL_GAP is the control gap. When the absolute value of error is less than NL_GAP, KNL = NL_MINMOD.

NL_TBAND is the transition band over which KNL is linearly adjusted as a function of error.

NL_HYST is a hysteresis value. Until absolute value of error exceeds NL_GAP + NL_HYST, KNL = NL_MINMOD. Once absolute value of error has exceeded NL_GAP + NL_HYST, absolute value of error must return to a value less than NL_GAP before KNL returns to a value of NL_MINMOD. If NL_GAP is 0, then the value of NL_HYST has no meaning (effectively assumed to be 0).

You can select the specifics of block execution by configuring I/O selection, signal conversion and filtering, feedforward calculations, tracking variables, setpoint and output limiting, PID equation structures, and block output action. The mode of the block determines setpoint and output selection.

I/O Selection

When you configure the PID function block, you select whether the source of the input value is a wired function block connection or a process input channel.

Input from Another Block – When you want the source to be an input from another function block, the input source (usually another block's output value) is connected to the IN connector on the PID function block. With a fieldbus extension, the connection to IN must be from another function block.

Input from a Process Input Channel – When you want the source to be an input from a process input channel, you configure the Device Signal Tag (DST) of the desired channel in the IO_IN parameter. There is no IO_IN parameter in the fieldbus extension.

Note When IO_IN is configured and IN is connected, the I/O input channel referenced by IO_IN takes precedence and IN is ignored.

You can configure anti-aliasing filtering, NAMUR limit detection, and overrange/underrange detection for the channel parameters. For information on these capabilities, refer to the I/O Configuration topics.

Simulation

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block.

During configuration, decide whether you want the simulated value/status to be entered manually into the function block or whether it will be supplied by another block.

When the value is entered manually:

- The operator first enables simulation by selecting the SIMULATE parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.
- When SIMULATE_IN is not connected (status = Bad: NotConnected), the operator enters the value to be used in the SIMULATE parameter Simulate Value field. In online operation, the operator can enter a simulated status value in the Simulate Status field.

Note Make sure SIMULATE_IN is not connected if you want to enter the value or status manually. When SIMULATE_IN is connected, the value entered in the SIMULATE_IN Simulate Value field is used as the simulated value.

When the value/status from another block is used:

- During configuration, connect SIMULATE_IN to the desired block output or parameter. Do not enter a value in the Simulate Value field of the SIMULATE_IN input; the block uses the connected value automatically.
- During operation, the operator enables simulation by selecting the SIMULATE parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.

Note Do not enter a value for the SIMULATE_IN parameter. If you enter a value and the status of SIMULATE_IN is not Bad: NotConnected, the value entered for SIMULATE_IN overrides any value you enter in SIMULATE.

There is no SIMULATE_IN in fieldbus.

Signal Conversion

Choose direct, indirect, or indirect square root signal conversion with the linearization type (L_TYPE) parameter:

Direct signal conditioning – simply passes through the value accessed from the I/O channel (or the simulated value when simulation is enabled).

Indirect signal conditioning – converts the accessed channel input value (or the simulated value when simulation is enabled) to the range and units of the PV parameter (PV_SCALE).

Indirect square root signal conditioning – converts the accessed channel input value (or the simulated value when simulation is enabled) by taking the square root of the value and scaling it to the range and units of the PV parameter (PV_SCALE).

You can view the accessed value (in percent) through the FIELD_VAL parameter.

When the converted input value is below the limit specified by the LOW_CUT parameter and the Low Cutoff I/O option (IO_OPTS) is enabled (True), a value of 0.0 is used for the converted value (PV). This option can be useful with zero-based measurement devices such a flowmeters.

You can choose to reverse the range for conversion to account for fail-open actuators by selecting the following I/O option:

Increase to Close – This option has an impact when a device signal tag is configured in IO_OUT. Increase to Close causes the milliamp signal on the analog output channel to be inverted in Man mode (and in Auto mode). That is, a full scale value on OUT will result in 4 mA on the channel. When IO_OUT is configured, the OUT value is the implied valve position and is not inverted when Increase to Close is True.

Note You can set I/O options in Manual or Out of Service mode only.

For complete descriptions of the supported I/O options, refer to the I/O Options topic.

Feedforward Calculation

You can activate the feedforward function with the FF_ENABLE parameter. When FF_ENABLE is True, the feedforward value (FF_VAL) is scaled (FF_SCALE) to a common range for compatibility with the output scale (OUT_SCALE). A gain value (FF_GAIN) is applied to achieve the total feedforward contribution.

Tracking

You can specify output tracking with control options and parameters. You can set control options in Out of Service mode only.

The Track Enable control option (CONTROL_OPTS) must be True for the track function to operate. When the Track in Manual control option is True, tracking can be activated and maintained when the block is in Man mode. When Track in Manual is False, tracking is disabled in Manual mode. Activating the track function causes the block's actual mode to go to Local Override (LO).

The tracking value parameter (TRK_VAL) specifies the value to be converted and tracked into the output when the track function is operating. The track scale parameter (TRK_SCALE) specifies the range of TRK_VAL.

When the track control parameter (TRK_IN_D) is True and the Track Enable control option is True, the TRK_VAL input is converted to the appropriate value and output in units of OUT_SCALE.

Setpoint and Output Limit Constraints

As part of download the output high and low limits are set to the configured values. If these limits have not been configured OUT_HI_LIM will be set to OUT_SCALE_HI and OUT_LO_LIM will be set to OUT_SCALE_LO.

The following constraints apply to download or direct entry:

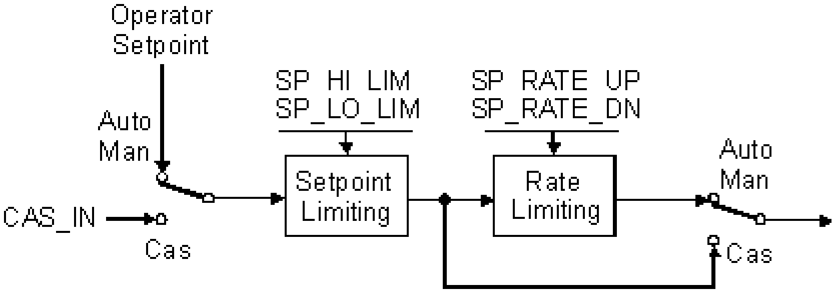
- SP_HI_LIM is restricted to PV_SCALE_HI+.1*(PV_SCALE_HI-PV_SCALE_LO).
- SP_LO_LIM is restricted to PV_SCALE_LO-.1*(PV_SCALE_HI-PV_SCALE_LO).
- OUT_HI_LIM is restricted to OUT_SCALE_HI+.1*(OUT_SCALE_HI-OUT_SCALE_LO).
- OUT_LO_LIM is restricted to OUT_SCALE_LO-.1*(OUT_SCALE_HI-OUT_SCALE_LO).

If the new scale causes a limit to be outside of these rules, the DeltaV system forces the limit within the rules.

The SP or OUT parameters are not changed as a result of changing the scale or limits. However, if OUT violates the new limits OUT will be forced within the limits on the next pass.

Setpoint Selection and Limiting

Setpoint selection is determined by the mode. The following diagram shows the method for setpoint selection:



PID Function Block Setpoint Selection

You can limit the setpoint by configuring the SP_HI_LIM and SP_LO_LIM parameters. In Cas mode, the setpoint comes from the CAS_IN input. In Auto mode, the setpoint is adjusted by the operator. You can limit the setpoint rate of change by configuring the SP_RATE_UP and SP_RATE_DN parameters.

Output Selection and Limiting

Output selection is determined by mode. In Auto, Cas, and RCas modes, the output is computed by the PID control equation. In Man and ROut modes, the output can be entered manually.

Note If the IO_OUT parameter is defined for direct output, use any connection to OUT for calculation purposes only (for example, an input to a calculation block).

You can limit the output by configuring the OUT_HI_LIM and OUT_LO_LIM parameters.

Bumpless Transfer and Setpoint Tracking

You can select setpoint tracking by configuring the following control options (CONTROL_OPTS):

SP-PV Track in LO or IMan

SP-PV Track in Man

SP-PV Track in ROut

When one of these options is set, the SP value is set to the PV value while in the specified mode.

You can select the value that an upstream controller uses for bumpless transfer and reset limiting by configuring the following control option:

Use PV for BKCAL_OUT

When this option is not selected, the working setpoint (SP_WRK) is used for BKCAL_OUT.

With the Use PV for BKCAL_OUT control option, the BKCAL_OUT value tracks the PV value. A master controller whose BKCAL_IN parameter receives the slave PID block's BKCAL_OUT in an open cascade strategy forces its OUT to match BKCAL_IN, thus tracking the PV from the slave PID block. If the master is another PID, then if the master PID has selected Dynamic_Reset_Limiting in FRSIPID_OPTS the external reset portion of the master PID now uses the PV of the secondary as its input. This provides an automatic adjustment of reset action in the master based on the performance of the slave.

You can set control options in Out of Service mode only.

PID Equation Structures

Parameter STRUCTURE in the PID is used to select which of the three PID actions (Proportional, Integral and Derivative) are active and how the actions are applied.

You can select the PID equation structure to apply controller action by configuring the STRUCTURE parameter. Select one of the following choices:

PID Action on Error — Proportional, integral and derivative action are applied to the error (SP - PV). If RATE is non-zero, a setpoint change will exhibit both a proportional and derivative kick. This structure is typically used to get fastest possible setpoint response when derivative action is used (RATE>0) and RATE is not so large as to make the resultant kick too great. A small SPFILTER value or SP RATE limiting can be used to reduce the worst case kick.

PI Action on Error, D Action on PV — Proportional and integral action are applied to error; derivative action is applied to PV. A setpoint change will exhibit a proportional kick.

I Action on Error, PD Action on PV — Integral action is applied to error; proportional and derivative action are applied to PV. There is no proportional or derivative kick on a setpoint change.

PD Action Error — Proportional and derivative actions are applied to error; there is no integral action. This structure will result in a steady state offset of PV from SP; the size of the offset will be determined by GAIN of the PID and the process gain. Offset is typically adjusted with BIAS of the PID. A setpoint change will exhibit a proportional and derivative (if RATE>0) kick.

P Action on Error, D Action on PV — Proportional action is applied to error; derivative action is applied to PV; there is no integral action. This structure will result in a steady state offset of PV from SP; the size of the offset will be determined by GAIN of the PID and the process gain. Offset is typically adjusted with BIAS of the PID. A setpoint change will exhibit a proportional kick.

ID Action on Error — Integral and derivative action are applied to error; there is no proportional action. This structure is typically selected for use in integral-only applications (RATE=0). It is also used in cases where the process exhibits the tendency to first move in the opposite direction from its final steady state value. There is a derivative kick on an SP change.

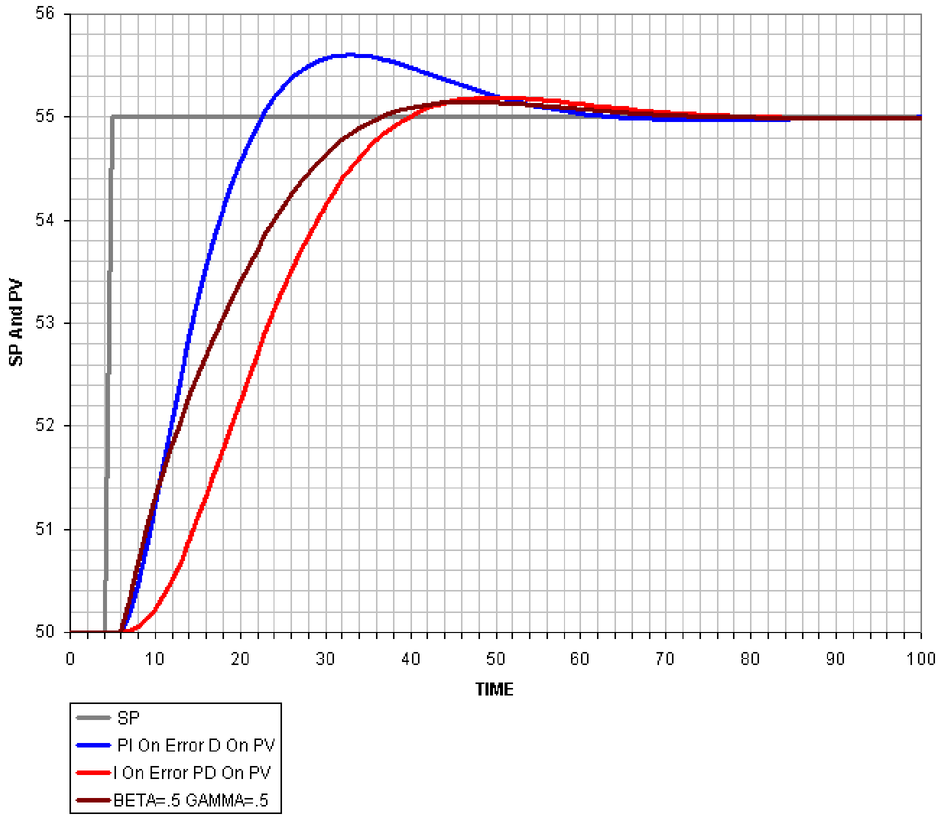
I Action on Error, D Action on PV — Integral action applied to error; derivative action applied to PV; there is no proportional action. This structure is typically selected for use in integral-only applications (RATE=0). It is also used in cases where the process exhibits the tendency to first move in the opposite direction from its final steady state value. There is no derivative kick on an SP change.

Two Degrees of Freedom Controller — Two parameters (BETA and GAMMA) can be adjusted to determine the degree of proportional (BETA) action and derivative (GAMMA) that will be applied to SP changes. The range for BETA and GAMMA is 0-1. BETA=0 means no proportional action is applied to SP change. BETA=1 means full proportional action is applied to SP change. GAMMA=0 means no derivative action applied to SP change. GAMMA=1 means full derivative action is applied to SP change. For values greater than 0 and less than 1, the number represents the decimal fraction of the action applied to SP change. This structure then can be used to get any of the structures that include all three (actions) with adjustable action on SP changes for proportional and derivative action.

Often, when tuning a control loop for disturbance rejection, the setpoint response exhibits considerable overshoot. This is particularly true when there is derivative action required and the derivative action is taken only on PV (to avoid large bumps in output as the result of modest setpoint changes). The Two Degrees of Freedom structure provided by the DeltaV system allows shaping of the setpoint response by adjusting the proportional and derivative action applied to setpoint. The adjustment parameters are BETA (for proportional) and GAMMA (for derivative). Tuning range is from no action to full action (0 to 1).

Two Degrees of Freedom is selected with the STRUCTURE parameter. The following figure illustrates the setpoint response for a loop tuned for good disturbance rejection with little or no overshoot in the disturbance response.

Adjustment of BETA and GAMMA can significantly reshape the setpoint response and drastically reduce the overshoot from that of a PID that has full proportional and no derivative action on setpoint.



SP Response for Different Structures

When Use Nonlinear Gain Modification is selected in FRSIPID_OPTS, proportional action (if called for by STRUCTURE) is applied to error and standard form equation is applied.

Reverse and Direct Action

You can select the block output action by configuring the Direct Acting control option.

Note You can set control options in Out of Service mode only.

Reset Limiting

The PID function block provides a selection between clamped integral action or dynamic reset limiting (external reset) that prevents windup when output or input constraints are encountered. By selecting Dynamic_Reset_Limiting in FRSIPID_OPTS a master PID in a cascade uses the BKCAL_OUT of the slave block. If the slave block passes back PV then the reset action dynamically tracks the PV of the slave. The Dynamic_Reset_Limiting option is what you will want to use in most control selector applications.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Out of Service – The block is in Out of Service (OOS) mode.

Readback failed – The I/O readback failed.

Output failure – Either the output hardware or the DST is bad.

Local Override – The block is in Local Override (LO) mode.

Simulate active – Simulation is enabled, and the block is using a simulated value in its execution.

Input failure/process variable has Bad status – The source of the block's process variable is bad. Indicates a bad status on a linked IN parameter, a hardware failure (if the block directly references DSTs) a non-existent Device Signal Tag (DST), or a Bad status on the SIMULATE parameter.

Modes - PID Function Block

The PID function block supports eight modes:

Out of Service (OOS)

Initialization Manual (IMan)

Local Override (LO)

Manual (Man)

Automatic (Auto)

Cascade (Cas)

Remote Cascade (RCas)

Remote Out (ROut)

You can configure the OOS, Man, Auto, Cas, RCas and ROut modes as permitted modes for operator entry.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Alarm Detection - PID Function Block

Block alarm detection is based on the PV and SP values. You can configure the following alarm limits to compare to the PV value for alarm detection:

- High (HI_LIM)
- High high (HI_HI_LIM)
- Low (LO_LIM)
- Low low (LO_LO_LIM)

You can configure the following alarm limits to compare to the difference between the SP and PV values (process error) for deviation alarm detection:

- Deviation high (DV_HI_LIM)
- Deviation low (DV_LO_LIM)

Note Deviation alarms are suppressed on SP changes. When the PV comes within the deviation limits, the deviation alarm is enabled again.

Status Handling - PID Function Block

The actual block mode of the PID block goes to Man if the PV status is Bad. Thorough the use of STATUS_OPTS, you can determine what conditions will cause the PV status to be BAD.

When the PID is configured to directly access an input channel for its input, the status of the channel input is determined as described for the Analog Input block. For more information, refer to the Status Handling - Analog Input Function Block topic. Otherwise, the IN parameter status is used as the block input status.

Processing of the input status within the PID block to determine the PV status may be modified by your choice of the Status options (STATUS_OPTS). The options are:

- **Bad if Limited**
- **Uncertain if Limited**
- **Target to Manual if Bad IN**
- **Use Uncertain as Good**

Note that the options available in a fieldbus block are different:

- **Target to Manual if Bad IN**
- **Use Uncertain as Good**
- **IFS if Bad CAS_IN**
- **IFS if BAD IN**

Note You can set the status options in Out of Service mode only.**Note** You can set the status options in Out of Service mode only.

If the input status is limited, the PV status is set to Bad or Uncertain when you select the Bad if Limited or Uncertain if Limited options respectively. Similarly, if the status of the input is Uncertain, then the PV status is set to Good or Bad based on your selection of enabled or disabled for the Use Uncertain as Good option.

Note When the option Uncertain if Limited is selected, then as long as the input status is limited, the PV status will be set to Uncertain (even when the input status is Bad). Selecting this option and the Use Uncertain as Good disables the mode from changing to Man under these condition when the target mode is Auto.

If you select the Target to Manual if Bad IN status option and the PV status evaluates to Bad during operation, the target mode and the actual mode are set to Man. If the status of the PV changes back to Good, the target mode (and actual mode) remain in Man.

Parameters - PID Function Block

The following table lists the system parameters for the PID function block:

PID Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active).
ALARM_HYS	Percent	The amount the alarm value must return within the alarm limit before the associated active alarm condition clears. ALARM_HYS is limited to 50% of scale.
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
ALPHA	None	The filter factor for derivative action. The default value is 0.125. The valid range in run time is 0.05 to 1.0. Increasing ALPHA increases damping of derivative action.
ARW_HI_LIM	OUT	High limit of Anti-Reset Windup. When the output is beyond ARW_HI_LIM and the integral action is returning toward the limit, then the applied RESET time is reduced by a factor of 16. ARW_HI_LIM must be set within the limits of OUT_HI_LIM and OUT_LO_LIM.
ARW_LO_LIM	OUT	Low limit of Anti-Reset Windup. When the output is beyond ARW_LO_LIM and the integral action is returning toward the limit, then the applied RESET time is reduced by a factor of 16. ARW_LO_LIM value must be set between OUT_HI_LIM and OUT_LO_LIM.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	The time over which an internal balancing bias will be dissipated. Only has practical meaning when the STRUCTURE parameter is a P + D selection.

Parameter	Units	Description
BETA	None	Fraction of proportional action applied to SP change. For a value of 0.6, 60% of the proportional action applied to SP change. The value of BETA can be changed over a range of 0-1 if STRUCTURE is set to Two Degrees of Freedom Control. Otherwise, it is automatically set to a value of 1 or 0 based on the Structure selection.
BIAS	None	The bias value.
BKCAL_IN	EU of OUT_SCALE	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BKCAL_OUT	EU of PV_SCALE	The value and status required by the BKCAL_IN input of another block to prevent reset windup and provide bumpless transfer to closed loop control.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the PID function block are Simulate active, Local override, Input failure/process variable has *Bad* status, Output failure, Readback failed, and Out of Service.
BYPASS	None	When enabled and the block is in AUTO mode, bypasses the normal control algorithm by transferring the SP value (in percent) to OUT. When disabled, the block operates normally. To turn BYPASS on or off, select the CONTROL_OPTS Bypass Enable option and set the block to MAN mode.
CAS_IN	EU of PV_SCALE	The remote analog setpoint value from another block.
CONTROL_OPTS	None	Control options. Allows you to specify control strategy options. The supported control options for the PID function block are No Out Limits in Manual, Obey SP Limits if Cas or RCas, Act on IR, Use PV for BKCAL_OUT, Track in Manual, Track Enable, Direct Acting, SP Track Retained Target, SP-PV Track in LO or IMan, SP-PV Track in ROut, SP-PV Track in Man, and Bypass Enable.
DV_HI_ACT	None	The result of alarm detection associated with DV_HI_LIM. If DV_HI_ACT equals True, DV_HI_LIM has been exceeded.
DV_HI_LIM	EU of PV_SCALE	The amount by which PV can deviate above SP before the deviation high alarm is triggered. When this limit is exceeded, DV_HI_ACT is set to True.
DV_LO_ACT	None	The result of alarm detection associated with DV_LO_LIM. If DV_LO_ACT equals True, DV_LO_LIM has been exceeded.

Parameter	Units	Description
DV_LO_LIM	EU of PV_SCALE	The amount by which PV can deviate below SP before the deviation low alarm is triggered. When this limit is exceeded, DV_LO_ACT is set to True. Note that DEV_LO_LIM is a negative number and is compared against (PV - SP).
ERROR	EU of PV_SCALE	The difference between SP (setpoint) and PV (process variable).
FF_ENABLE	None	Enables/disables feedforward control.
FF_GAIN	None	The feedforward gain value. FF_VAL is multiplied by FF_GAIN before it is added to the calculated control output.
FF_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the feedforward value (FF_VAL).
FF_VAL	EU of FF_SCALE	The feedforward control input value and status.
FIELD_VAL	Percent	The value and status from the I/O card or from the simulated input if simulation is enabled.
FORM	None	Selects equation form (series or standard). If Use Nonlinear Gain Modification is selected in FRSIPID_OPTS, the form automatically becomes standard, regardless of the configured selection of FORM.
FRSIPID_OPTS	None	FRSI add-on control options. Supported options are Dynamic Reset Limiting, Use Nonlinear Gain Modification.
GAIN	None	The normalized proportional (multiplier) gain value.
GAMMA	None	Fraction of derivative action taken on SP. For a value of 0.6, 60% of the derivative action is applied to SP. The value of GAMMA can be changed over a range of 0-1 if STRUCTURE is set to Two Degrees of Freedom Control. Otherwise, it is automatically set to a value of 1 or 0 based on the Structure selection.
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.
HI_HI_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the high high alarm condition.
HI_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the high alarm condition.

Parameter	Units	Description
IDEADBAND	EU of PV_SCALE	The dead band value. When the error gets within IDEADBAND, the integral action stops. The proportional and derivative action continue.
IN	EU of PV_SCALE	The analog input value and status (for example, the connection for the PV input from another block).
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 only if the following conditions are true: The Write to Inspect Alarm context menu item has been selected from Inspect for this block. With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition exists for Variability only if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
IO_IN	None	Defines the input DST for the I/O channel used for the PV.
IO_OPTS	None	I/O options. Allows you to select how the I/O signals are processed. The supported I/O options for the PID block are Increase to Close and Low Cutoff.
IO_OUT	None	Defines the output DST for the block.
IO_READBACK	None	Defines the Device Signal Tag (DST) for the input channel that provides readback for the value written to the channel defined by IO_OUT.
L_TYPE	None	Linearization type. Determines whether the field value is used directly (Direct), is converted linearly (Indirect), or is converted with the square root (Indirect Square Root).
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.
LO_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.
LO_LO_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the low low alarm condition.
LOW_CUT	EU of PV_SCALE	Activated when the Low Cutoff I/O option is enabled. When the converted measurement is below the LOW_CUT value, the PV is set to 0.0.

Parameter	Units	Description
MODE	None	Parameter used to show and set the block operating state. MODE contains the actual, target, permitted, and normal modes.
NL_GAP	EU of PV Scale	The configured range of ERROR, positive or negative, where the gain modifier is at a minimum value. The range is 0->(PV_SCALEHI-PV_SCALELO).
NL_HYST	EU of PV Scale	Gap action hysteresis value. The range is 0->(PV_SCALEHI-PV_SCALELO).
NL_MINMOD	None	The configured minimum gain modifier. The range is 0->1.0.
NL_TBAND	EU of PV Scale	The configured range of ERROR, positive or negative, where the gain modifier transitions between NL_MINMOD and 1.0. The range is 0->(PV_SCALEHI-PV_SCALELO).
OUT	EU of OUT_SCALE	The analog output value and status.
OUT_HI_LIM	EU of OUT_SCALE	The maximum output value allowed.
OUT_LO_LIM	EU of OUT_SCALE	The minimum output value allowed.
OUT_READBACK	EU of OUT_SCALE	The value and status of the output channel referenced by IO_READBACK.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
PV	EU of PV_SCALE	The process variable used in block execution and alarm limit detection.
PV_FTIME	Seconds	The time constant of the first-order PV filter.
PV_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with PV.
RATE	Seconds	The derivative action time constant.
RCAS_IN	EU of SP_SCALE	The remote analog setpoint value and status. Input provided by a device or the output of another block.
RCAS_OUT	EU of PV_SCALE	The output provided for bumpless mode transfer and reset limiting by the source of RCAS_IN. Essentially the equivalent of BKCAL_OUT for RCAS_IN.
RESET	Seconds per repeat	The integral action time constant.
ROUT_IN	EU of OUT_SCALE	Remote output value and status. Input provided by a device to the control block for use as the output (ROut mode)

Parameter	Units	Description
ROUT_OUT	EU of OUT_SCALE	The output provided for bumpless mode transfer and reset limiting by the source of ROUT_IN. Essentially the equivalent of BKCAL_OUT for ROUT_IN.
SHED_OPT	None	Defines action to be taken on remote control device timeout.*
SHED_TIME	Seconds	The maximum allowable time between RCAS_IN or ROUT_IN updated. If exceeded, mode shedding takes place.
SIMULATE	Percent	Enables simulation and allows you to enter an input value and status. The SIMULATE value is used by the block only when SIMULATE_IN is not connected.
SIMULATE_IN	Percent	The input connector value and status used by the block instead of the analog measurement when simulation is enabled. If SIMULATE_IN is connected or has a manually entered value, SIMULATE_IN always overrides SIMULATE. **Note** When SIMULATE_IN is wired from an input source on the function block diagram, it always overrides a manually entered value in SIMULATE.
SP	EU of PV_SCALE	The block's setpoint value.
SP_FTIME	Seconds	Time constant of the first order SP filter.
SP_HI_LIM	EU of PV_SCALE	The highest SP value allowed.
SP_LO_LIM	EU of PV_SCALE	The lowest SP value allowed.
SP_RATE_DN	EU of PV_SCALE per second	Ramp rate at which downward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_RATE_UP	EU of PV_SCALE per second	Ramp rate at which upward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_WRK	EU of PV_SCALE	The working setpoint of the block subjected to SP_RATE_DN and SP_RATE_UP.

Parameter	Units	Description
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*
STATUS_OPTS	None	Status options. Allows you to select options for status handling and processing. The supported status option for the PID block depends on whether or not the block is in a fieldbus device. In a Fieldbus device: Target to Manual if Bad IN Use Uncertain as Good IFS if Bad CAS_IN IFS if BAD IN In a DeltaV Controller: Bad if Limited Uncertain if Limited Target to Manual if Bad IN Use Uncertain as Good
STDEV_CAP	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.
STDEV	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The standard deviation of PV. For analog control blocks in AUTO, mean is assumed to be the SP. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STRATEGY	None	The strategy filed can be used to help group blocks. This data is not checked or processed by the block.*
STRUCTURE	None	Defines PID equation structure to apply controller action.

Parameter	Units	Description
TRACK_OPT	None	Tracking Option. Allows you to select the tracking behavior when the status of the TRK_IN_D is Bad. The three tracking options are Always Use Value - The block reacts to the current value of TRK_IN_D regardless of the status. Use Last Good Value - The block uses the value of TRK_IN_D the last time its status was not Bad. This is the default value for TRACK_OPT. Track if Bad - If the status of TRK_IN_D is Bad, the block reacts as if the value if True, even if the value is False.
TRK_IN_D	None	Discrete input that initiates external tracking.
TRK_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the external tracking value (TRK_VAL).
TRK_VAL	EU of TRK_SCALE	The analog input used in the external tracking function.

* These parameters are only visible when the function block is located in a Fieldbus device.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - PID Function Block

The PID function block is a powerful, flexible control algorithm that is designed to work properly in a variety of control strategies. The PID block is configured differently for different applications. The following examples describe how to use the PID block for closed loop control: basic PID loop, feedforward control, cascade control with master and slave, complex cascade control with override, and PID control with tracking.

Closed Loop Control

Implement basic closed loop control by taking the error difference between the setpoint (SP) and the process variable (PV) and calculating a control output signal using a PID (Proportional Integral Derivative) function block.

Proportional control responds immediately and directly to a change in the PV or SP. The proportional term (GAIN) applies a change in the loop output based on the current magnitude of the error. With only the proportional term (GAIN), the control loop will likely have a steady state error.

Integral control eliminates steady state error. It integrates the error until it is negligible. The integral term (RESET) applies a correction based on the magnitude and duration of the error. Lowering RESET increases integral action.

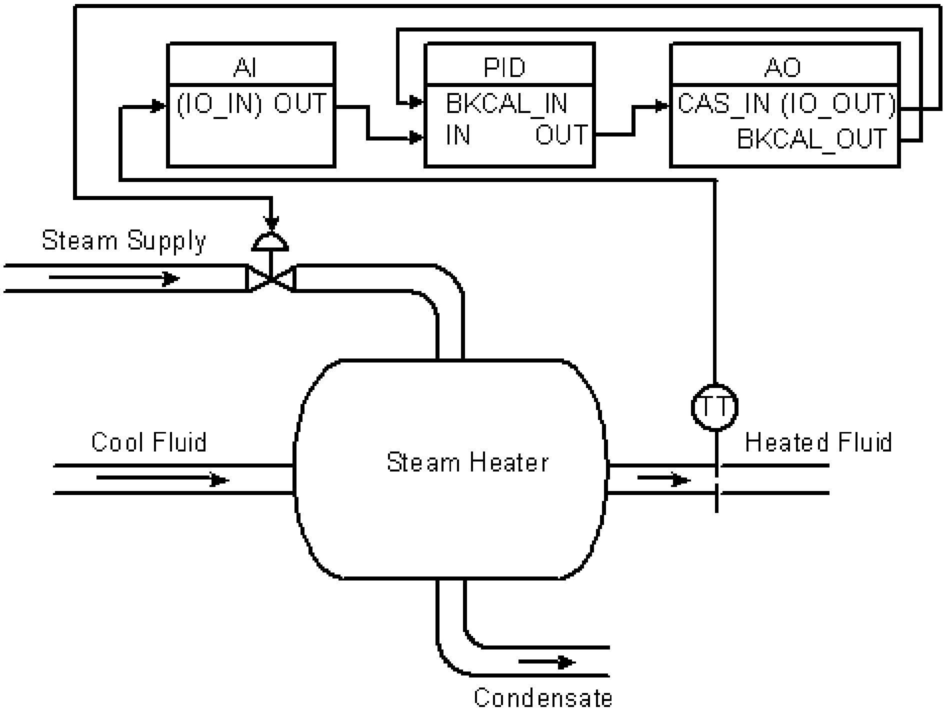
The derivative term (RATE) applies a correction based on the rate of change of error. Use derivative control where large measurement lags exist, which is typically temperature control.

The MODE parameter is a switch that indicates the target and actual mode of operation. Mode selection has a large impact on the operation of the PID block.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Application Example: **Basic PID Block for Steam Heater Control**

A process fluid is heated by steam in a heat exchanger, as in the following example:



Basic Setup for PID Steam Heater Control

In this example, the PID controller accepts the heated fluid temperatures as an input and provides a signal to the AO block, which sends the control signal to the steam feed valve.

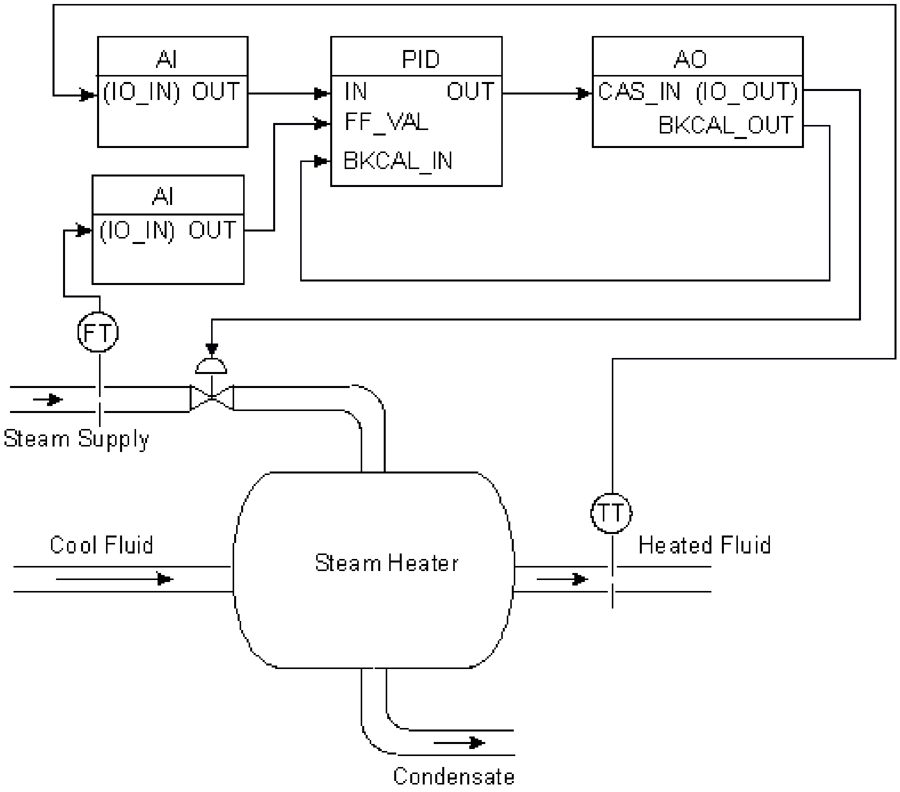
Note You can configure the PID function block by referencing the I/O directly and not using AI and AO function blocks. In this case, I/O channels are addressed with the parameters IO_IN and IO_OUT. Status information is communicated to the block when you define the card and channel parameters. The BKCAL communication between the PID and AO blocks is a necessary part of this example. For more information on this BKCAL communication, refer to the Advanced Topics - BKCAL Communications topic.

Application Example: **Feedforward Control**

In the above example, control problems can arise because of a time delay caused by thermal inertia between the two flow streams. If, for example, steam flow declines, it can take some time for this disturbance to cause a drop in the heated fluid's temperature.

Control can be improved by measuring this disturbance and reacting to it before it manifests itself at the temperature transmitter. In the following figure, steam flow is measured (FT). A feedforward signal is sent to the controller to augment the signal to the valve if flow drops or to lower this signal if steam flow rises. (This applies to a configuration where the higher the signal, the greater the valve opening.)

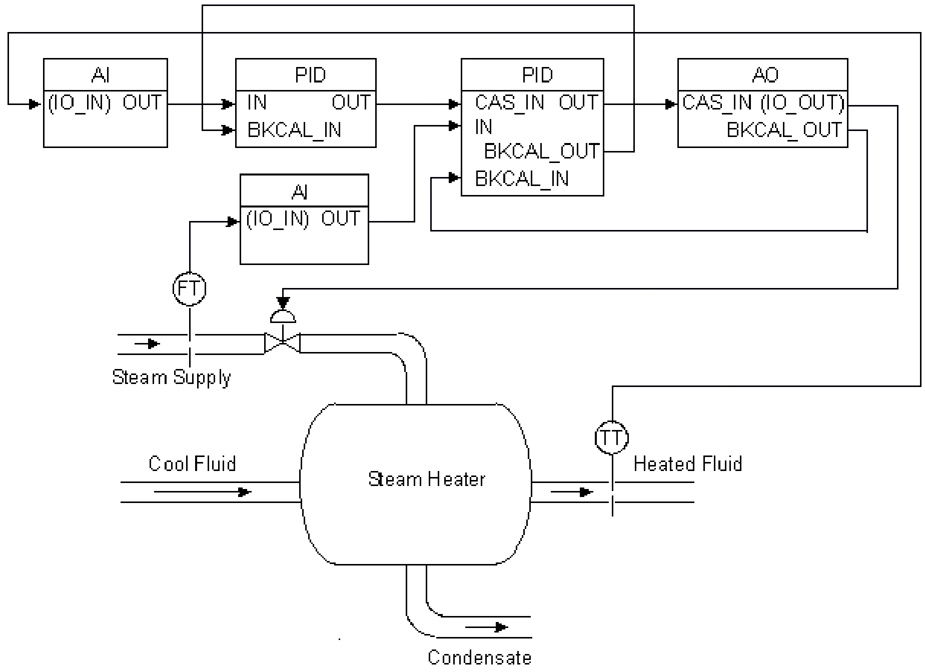
In this steam heater system, adding feedforward control improves the process outlet temperature response. The inlet steam flow is input to an AI function block and is connected to the FF_VAL connector on the PID block. Enable feedforward control (FF_ENABLE), scale the feedforward value (FF_SCALE), and apply a gain determined by tuning (FF_GAIN). The following figure shows the process and function block configuration for feedforward control:



PID Function Block Feedforward Control Example

Application Example: **Cascade Control with Master and Slave Loops**

The feedforward scheme in the above example requires that some correlation be predetermined between steam flow changes and the steam valve opening adjustments they make. Another way to deal with the time delay problem is to use cascaded controllers. This approach does not require finding a correlation between steam flow changes and their steam valve opening adjustments. In the cascade loop in the following figure, the output from the master temperature loop is used as the setpoint for the slave steam flow loop. The following diagram shows the process instrumentation for this example:



PID Function Block Cascade Control Example

The cascaded blocks shown can all be installed in a single module. Another method of configuring function blocks for this example is to put the temperature AI block and the master loop PID block in one module. The flow, slave PID block, and AO block are in a second module. Choose this method when you want to reference faceplates and alarms separately.

In the PID function block, BYPASS is used when the control function block is a slave block in a cascade. If a transmitter failure or some other problem occurs that causes the slave controller to see a Bad input signal, BYPASS can be activated so that the signal coming from the master controller passes through the slave to the field. In this manner, the loop still has some control; however, dynamic performance will suffer because a cascaded loop has effectively been replaced with a single PID.

Application Example: **Cascade Control with Override**

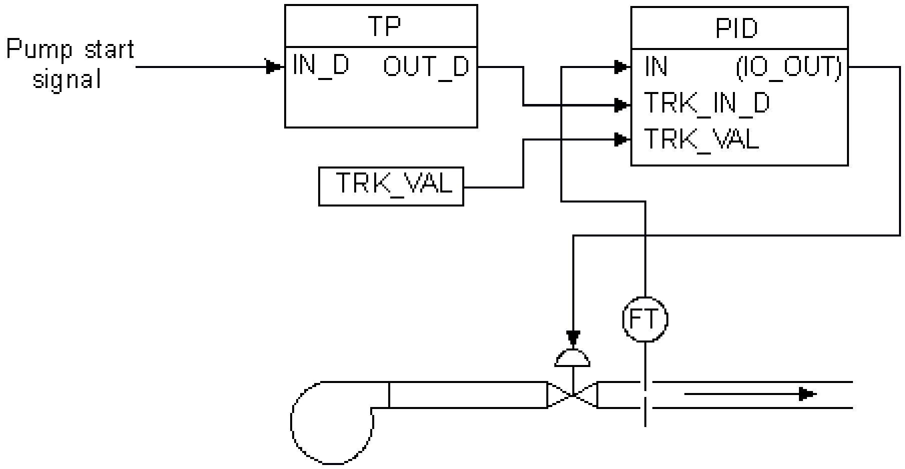
You might need override with cascaded PIDs. For more information on the use of override control, refer to Application Information - Control Selector Function Block.

Note Cascaded blocks use BKCAL_IN and BKCAL_OUT to pass statuses during cascade initialization and in limited states. Refer to Advanced Topics - BKCAL Communications for more information.

Application Example: **PID Control with Tracking**

An example where tracking is useful is a process operating outside of its normal operating range (for example, during a process startup).

In the following figure, a flow control valve is used to regulate the flow rate supplied by a pump. When the pump is started, the output flow from the pump is low for a short period while the pump is coming up to speed. The flow control valve and controller do not operate effectively at this low flow.

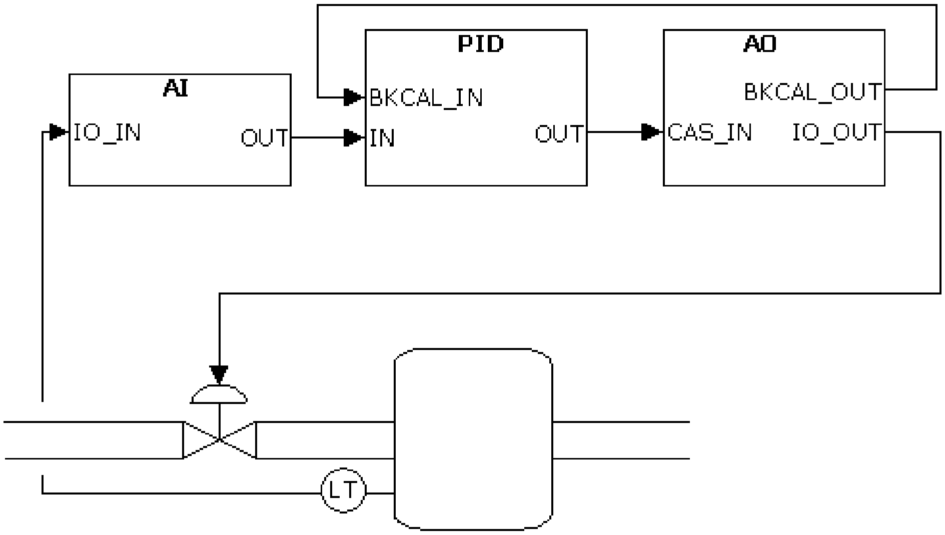


PID Control Using Tracking

Tracking can be used in this situation to set the valve at a predetermined opening for a given amount of time. The signal to start the pump is also directed through a timed pulse block to turn tracking on for the time duration specified in the timed pulse block. The tracking value is set at the needed valve opening. When the pump starts, tracking starts. The valve opens to the specified value for the duration of the timed pulse. At the end of the pulse, tracking is turned off, and the PID can initiate its normal control, regulating the valve to adjust the output flow.

Application Example: **Error-Squared Proportional Only Control Applied to Integrating Process**

A PID block, configured for proportional only control, is applied to a liquid level (for example, a surge tank level control might be performed this way).

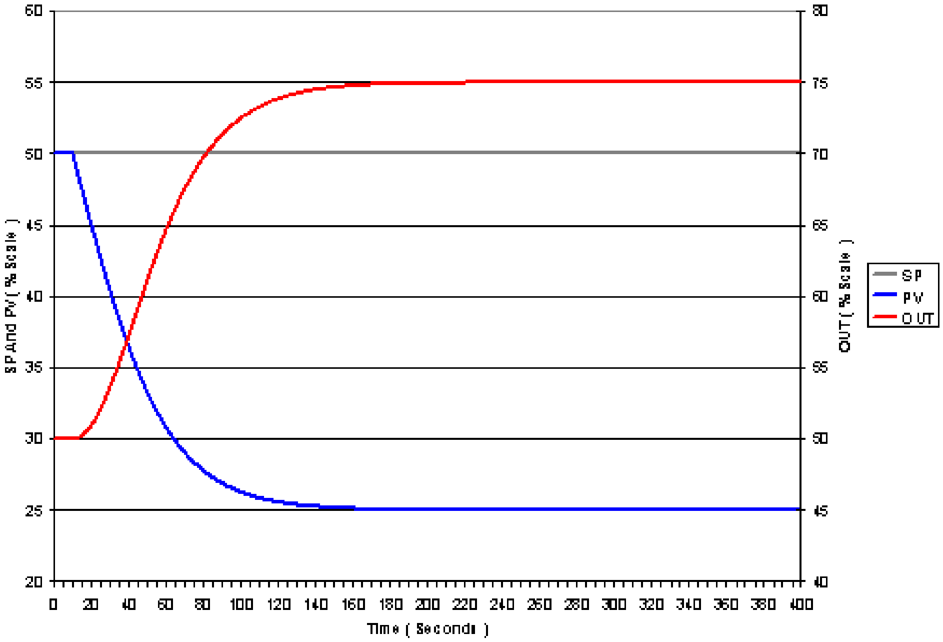


Proportional Only Control Applied to a Liquid Level

Use Nonlinear Gain Modification in FRSIPID_OPTS is selected.

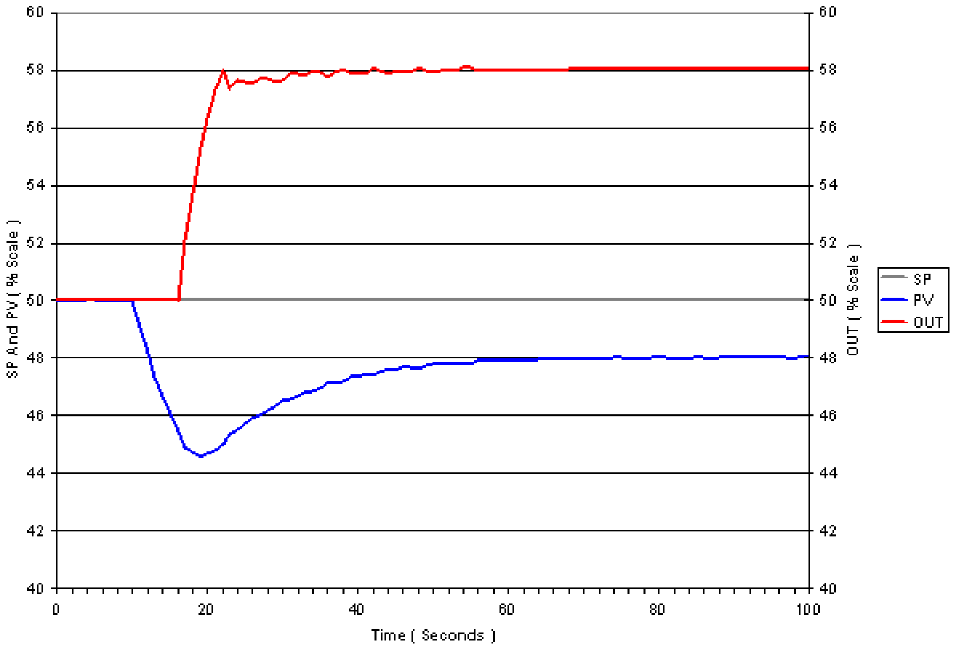
Gain=2; Reset=0; Rate=0; NL_GAP=0; NL_HYST=0; NL_TBAND=50; NL_MINMOD=0. This set of tuning parameters provides an error-squared output with a maximum gain of 2.

Starting balanced at 50% of scale (BIAS=50 %), a load disturbance equivalent to 25% of OUT_SCALE is introduced.



Proportional Error Squared Applied to Integrating Process

Application Example: **PID with Deadband, Transition Band**, **and Hysteresis**



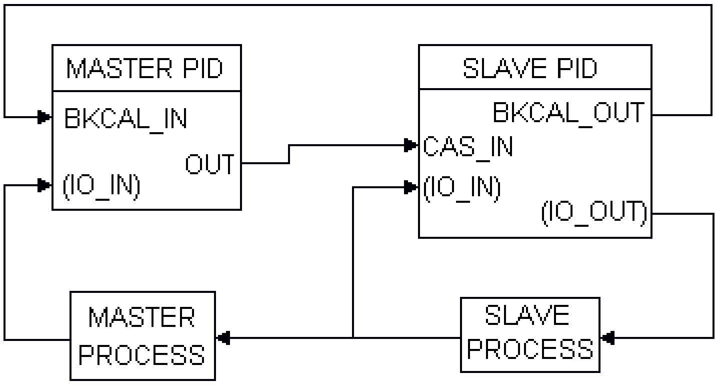
Control with Deadband, Transition Band, and Hysteresis

A self-regulating process is controlled with a PID configured for deadband, hysteresis and transition band. The block diagram of the loop is identical to that for the previous example. Tuning parameters are: NL_MINMOD=0; NL_GAP=2; NL_HYST=3; NL_TBAND=3; GAIN=2; RESET=7.5; RATE=0.

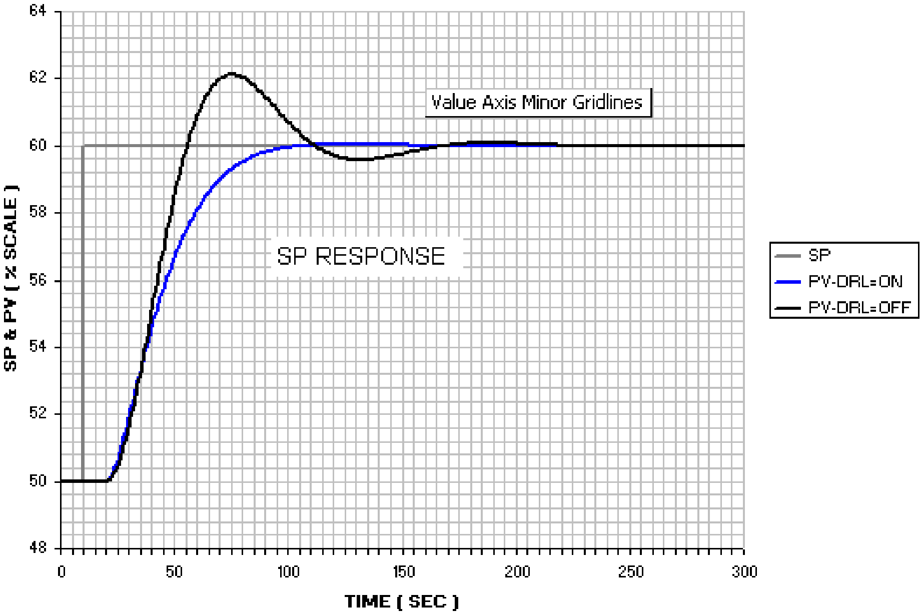
A disturbance equivalent to 25% of OUTSCALE is introduced at the process input. Until the error exceeds 5 (NL_GAP + NL_HYST), the output remains constant. Once the controller begins adjusting its output it continues to adjust until the error is brought to a value less than NL_GAP, at which point the output is held constant.

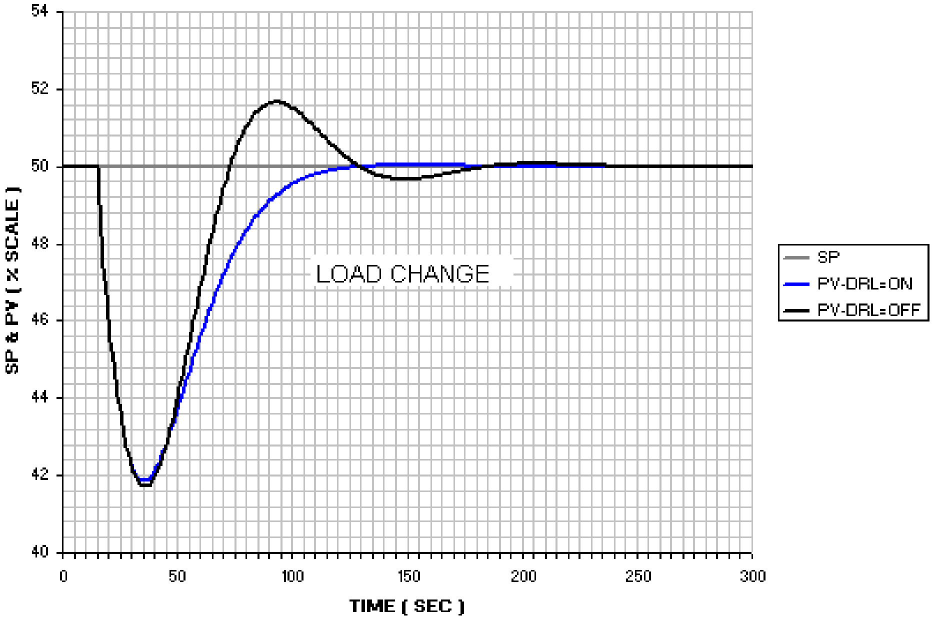
Application Example-Dynamic Reset Limiting In Primary of A Cascade

Two PID blocks are configured in a cascade control configuration. The MASTER PID is tuned for load response to approximately the best IAE (Integrated Absolute Error), without FRSI_PID_OPTS option Dynamic_Reset_Limiting (DRL) selected. The MASTER PID is then tuned to give the identical IAE with option Dynamic_Reset_Limiting selected. The response for SP and load responses are compared: Note that in this example the MASTER is using identical RESET and RATE, but in the case where the DYNAMIC_RESET_LIMITING option is selected the GAIN parameter is adjusted to about 1.6 the value applied in the case when option is not selected.



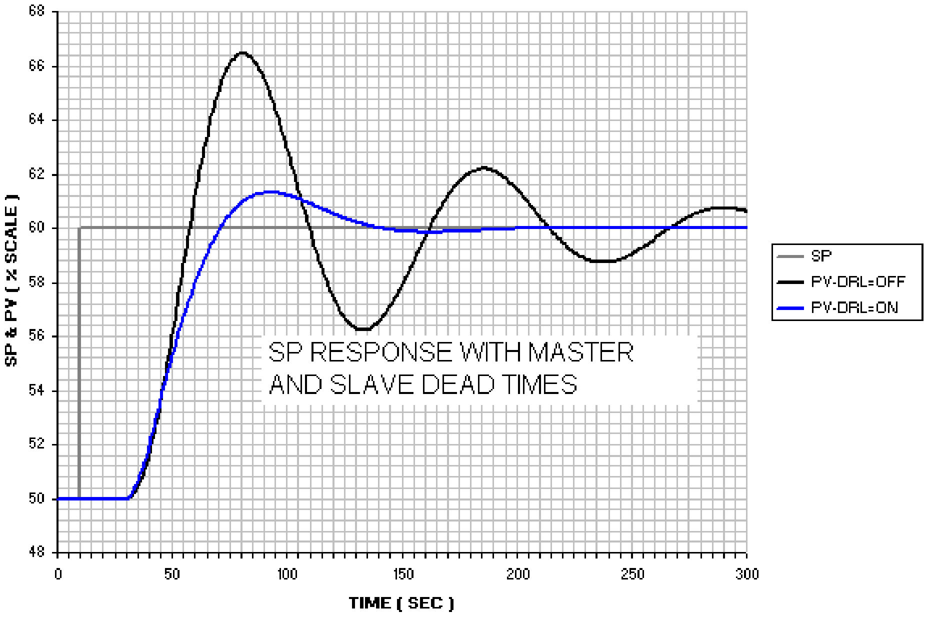
Dynamic Reset Limiting in Primary of a Cascade

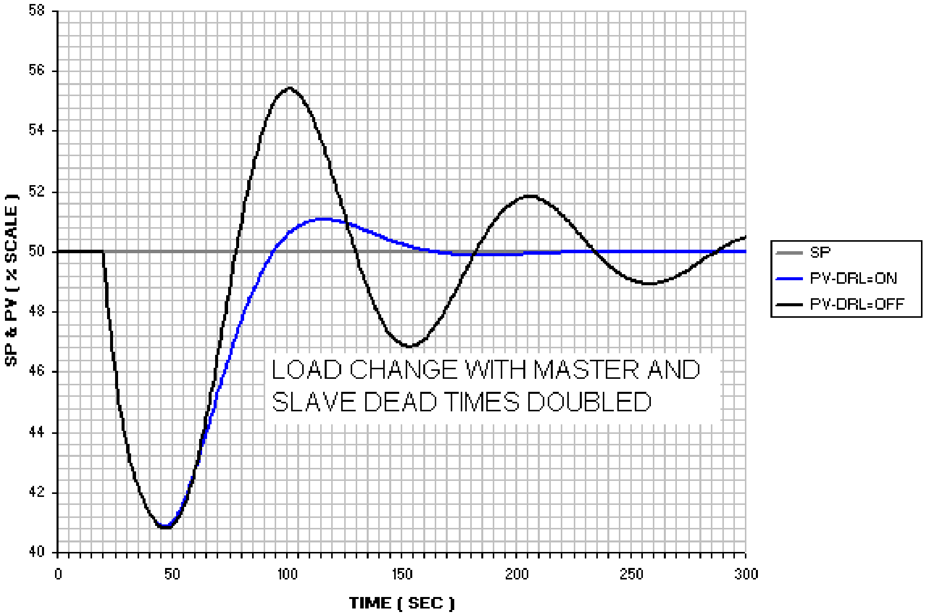




From the above plots it is apparent that master response is altered significantly by the use of Dynamic_Reset_Limiting. The response difference illustrated is typical. The tendency to overshoot is much reduced.

Both the MASTER and SLAVE processes are then modified by doubling the dead time in both.





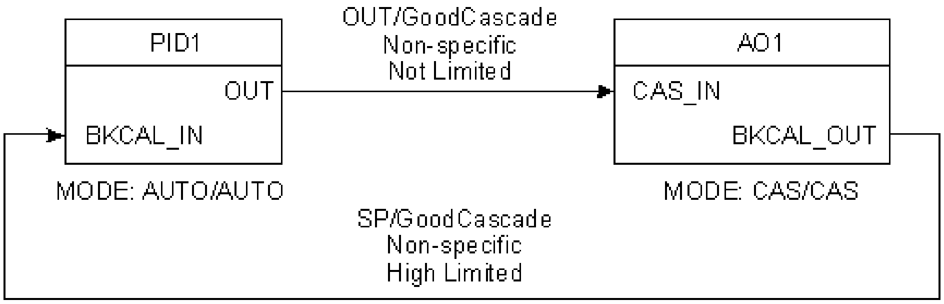
When the Dynamic_Reset_Limiting option is selected the responses are maintained at an acceptable level of performance, even though the GAIN in that case is higher. Not only does Dynamic_Reset_Limiting (or dynamic external reset) provide reset limiting in cases of limit violations, it also reduces the sensitivity of the control loop to changes in slave performance and stability, while providing much higher stability margins in the MASTER.

Advanced Topics - BKCAL Communications

Interblock Communication

The BKCAL parameters are used for interblock communication, specifically to let upstream blocks know the status of downstream blocks. Blocks that utilize the BKCAL parameters are referred to as cascaded blocks. When cascaded, the blocks act as a control unit that must coordinate its control activity.

The BKCAL parameters include a BKCAL_OUT parameter in a downstream block connected to a BKCAL_IN parameter in an upstream block. The downstream block passes a value (usually its setpoint) as well as a status to the upstream block. When the upstream block executes its algorithm, it takes into account the status of the downstream block. The most common use of BKCAL is to prevent reset windup. In other words, if a downstream block is in a limited state, it sends the limited status to the upstream control block, which turns off its integral action in response, as shown in the following figure.

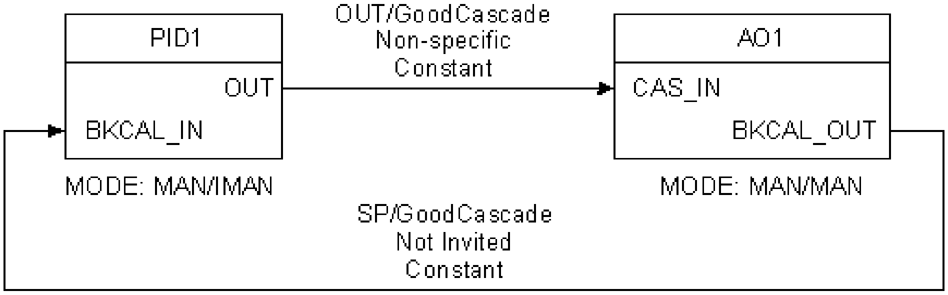


Anti-Reset Windup Protection with Control Cascade

Cascade Initialization

Another use of the BKCAL parameters is to initialize cascaded blocks (refer to Cascade Basics for more information). This occurs when the control cascade is automated (for example, when the blocks go from Manual to Auto or Cas mode). The simplest case of a control cascade is a master PID block linked to a slave AO block. The dialog that takes place between blocks during this process is illustrated in the following sequence of figures.

1 Start out both blocks with Man mode as the target modes.

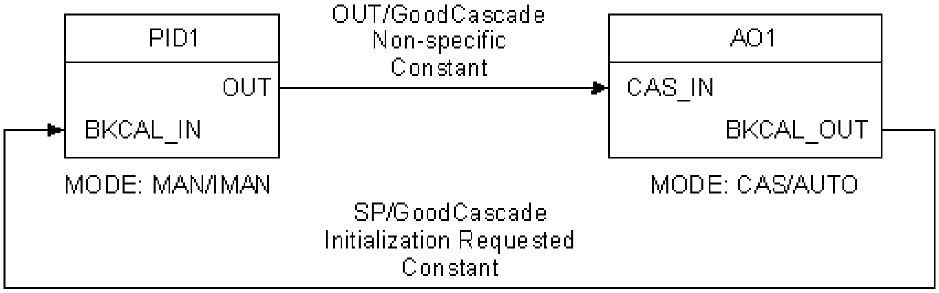


Blocks with Man Modes as Targets

In step 1, the cascade is open, that is, not automated. Both blocks have Manual as their target modes. Note that PID1 has IMan as its actual mode because the PID block senses that there is a slave block downstream, and the slave block is sending a Not Invited status signal up to the master block through the BKCAL port. Not Invited conveys to the master block that the slave is not in Cascade mode. Therefore, the cascade cannot be closed, and the master cannot operate as an automatic controller. Also, the slave block sends status information through its BKCAL_OUT parameter. This occurs because the status entering the CAS_IN parameter is GoodCascade rather than GoodNoncascade. If the status of CAS_IN were GoodNoncascade, the slave would conclude that the upstream block is not one that initializes as part of a control cascade.

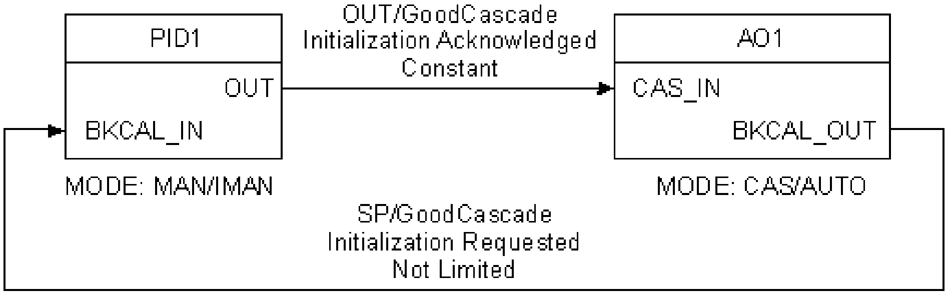
Close the cascade by first putting the slave block into Cascade mode. Set the target mode of AO1 to Cascade. This causes two things to happen: the actual mode of the AO block goes to Auto, and AO1/BKCAL_OUT sends out an initialization request to the upstream block, informing the upstream block that the downstream block is trying to go to Cascade mode. This is shown in step 2.

2 Set the downstream block's target mode to Cas.



Downstream Block Target Mode Set to Cas

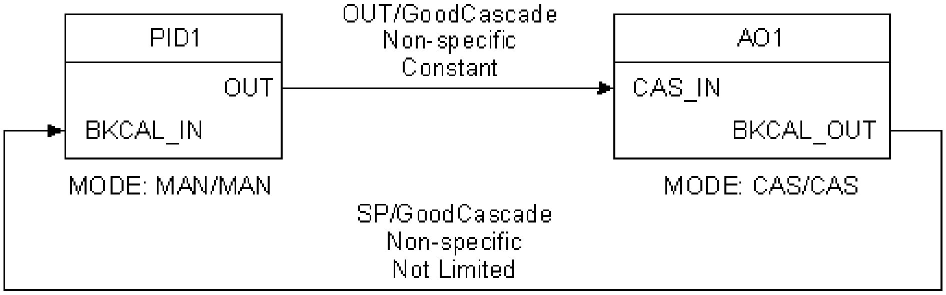
3 The upstream block initializes and notifies the downstream block that initialization has occurred.



Upstream Block Initialization and Notification of Downstream Block

In step 3, the master block has received the initialization request and initializes, that is, sets up SP for bumpless transfer. Then, the master block lets the slave block know that it has received the initialization request and successfully initialized. The process by which this occurs is an Initialization Acknowledged status sent through the master block's OUT parameter. If something were wrong with the upstream block (for example, it had tracking enabled or a bad input status), it would not be able to initialize and the cascade could not be closed. AO1 would be stuck in Cas/Auto mode, performing its function using the internal setpoint of the block.

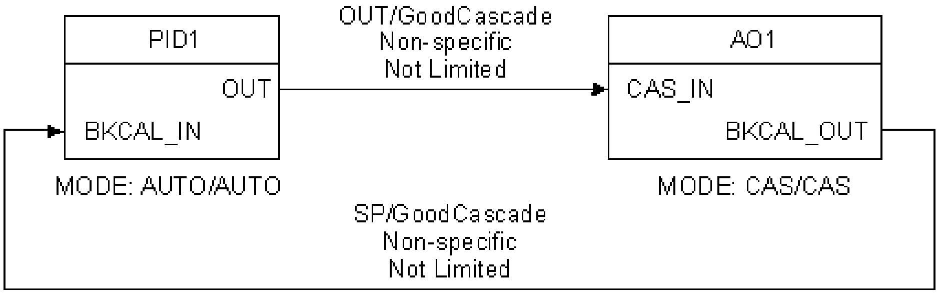
4 The downstream block changes to Cas mode and notifies the upstream block of this change.



Downstream Block Changes to Cas Mode and Notifies Upstream Block

Once the slave block receives the Initialization Acknowledged signal from the master block, the slave block's actual mode becomes Cas (step 4). This step closes the cascade. The slave block executes its algorithm using the SP supplied by the master block. At this point, the cascade is closed but not automated. The master PID is still in Man mode.

5 The upstream block sets and achieves Auto mode.



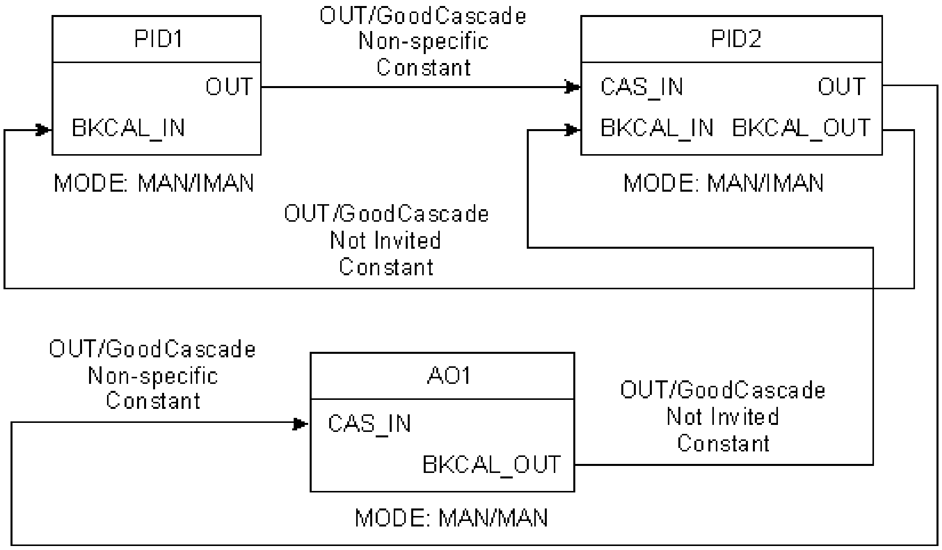
Upstream Block Sets and Achieves Auto Mode

Step 5 shows the fully automated cascade. The target mode of the master controller is set to Auto. Since PID1/BKCAL_IN has a status other than Not Invited (because the slave block is in Cas mode), the PID block is authorized to go to Auto mode. Note that if the user had set the target mode of PID1 to Auto at step 1, the block would have remained with mode Auto/IMan because of the Not Invited status received by the PID through BKCAL_IN.

Multi-level Cascade Initialization

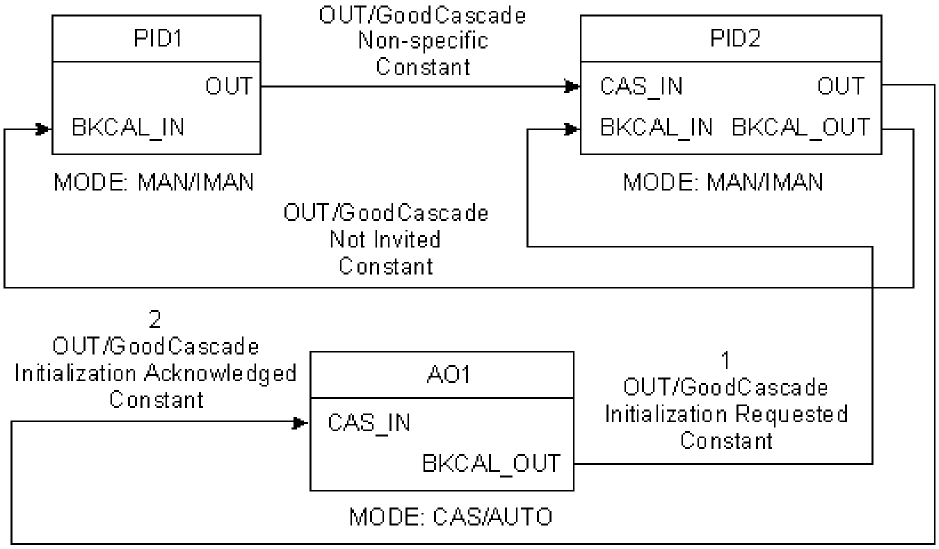
The following figure shows the initialization process for a multi-level cascade (that is, a cascade with two levels of cascading). This figure illustrates a control cascade with a downstream AO block. Cascade automation starts from the most downstream block and proceeds step by step upstream. When these steps are completed, all blocks in the cascade are in Cas mode except the master block, which is in Auto mode when the cascade is fully automated.

1 Start all blocks in Man mode.



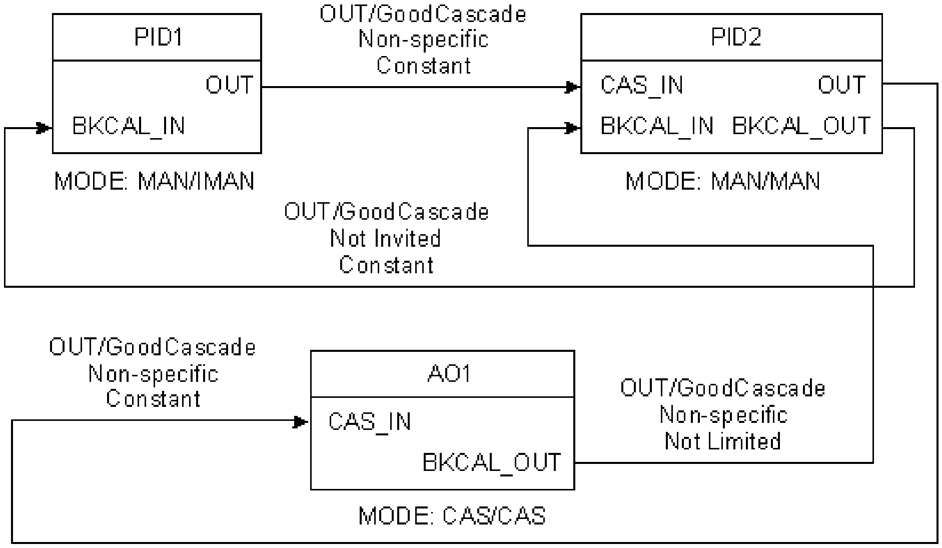
Blocks Start in Man Mode

2 AO1 sets the target mode to Cas; PID2 initializes and acknowledges.



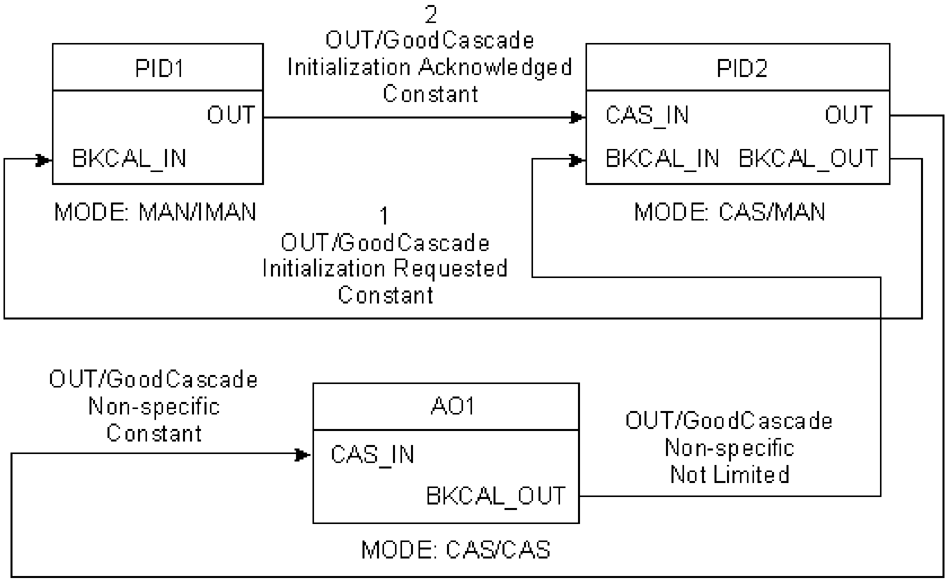
AO1 Setting of Target Mode to Cas; PID2 Initialization and Acknowledgement

3 AO1 achieves Cas mode; PID2 achieves Man mode.



AO1 Achievement of Cas Mode; PID2 Achievement of Man Mode

4 PID2 sets the target mode to Cas; PID1 initializes and acknowledges.



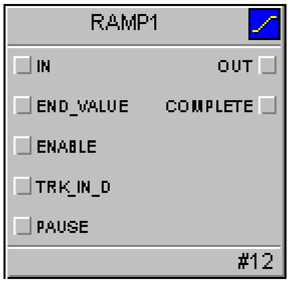
PID2 Setting of Target Mode to Cas; PID1 Initialization and Acknowledgement

The preceding examples regarding Cas mode also apply to RCas mode. The same communication takes place between the master and slave blocks. However, in RCas mode, substitute RCAS_IN for CAS_IN and RCAS_OUT for BKCAL_OUT.

Ramp Function Block

The Ramp function block creates a ramping output signal to increase or decrease a variable toward a specified target value at a defined rate. You define the time duration or rate of change of the ramp and the endpoint.

The Ramp function block supports signal status propagation. There are no modes or alarm detection in the block.



Ramp Function Block

IN is the analog input value and status that can be used as the ramp starting point.

END_VALUE is the ramp endpoint input value and status.

ENABLE activates the ramp calculation.

TRK_IN_D causes OUT to track IN.

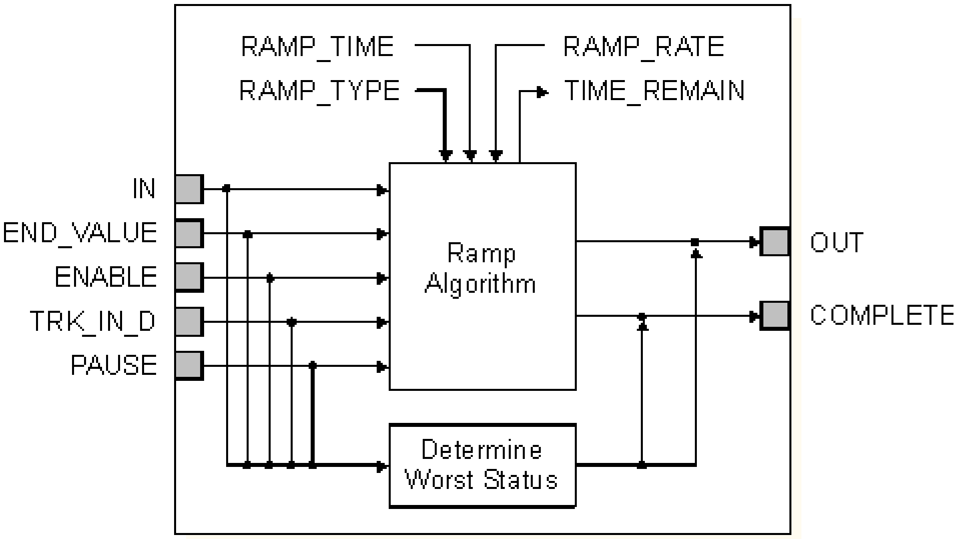
PAUSE stops the ramp calculation.

OUT is the output value and status that indicates a tracked or a ramped value.

COMPLETE indicates the ramp calculation is complete.

Schematic Diagram - Ramp Function Block

The following diagram shows the internal components of the Ramp function block:



Ramp Function Block Schematic Diagram

Block Execution - Ramp Function Block

The ramp begins when the ENABLE input transitions from False (0) to True (1).

- When the TRK_IN_D parameter is True (1), IN is the ramp starting value.
- When TRK_IN_D is False, the previous OUT value is the ramp starting value.

The END_VALUE parameter specifies the ramp endpoint and the target end value of OUT.

Specifying the Ramp Calculation

You specify the time duration of the ramping output with the RAMP_TIME parameter, or you specify the ramp rate directly with the RAMP_RATE parameter. The RAMP_TYPE parameter indicates whether to use RAMP_TIME or RAMP_RATE for the ramping calculation:

- When the RAMP_TYPE parameter is True (1), rate equals RAMP_RATE.
- When RAMP_TYPE is False (0), RAMP_TIME is used for the rate parameter calculation.

Ramp Execution

When the RAMP_TYPE, IN, or END_VALUE parameters change during the execution of the block, the ramp is reset with the new values. In addition, the ramp is reset when the selected RAMP_TYPE, RAMP_TIME, or RAMP_RATE parameter is changed during execution.

The TIME_REMAIN parameter is the time left until COMPLETE transitions to TRUE. The transition of the COMPLETE parameter to TRUE signals the end of the ramp calculation. The output will equal END_VALUE at this time.

The ramping OUT and TIME_REMAIN values can be stopped and held at their current value when the PAUSE parameter transitions to True. After PAUSE transitions from True to False, the block resets using the new

END_VALUE, IN, RAMP_TYPE, RAMP_TIME, or RAMP_RATE values. The block resets only if one or more of these values change.

When ENABLE is False, the COMPLETE and PAUSE parameters are set False and TIME_REMAIN is set to zero seconds.

The following table shows the relationship between the ENABLE, TRK_IN_D, PAUSE, COMPLETE, and OUT parameters in the Ramp function block:

Ramp Function Block Parameter Settings

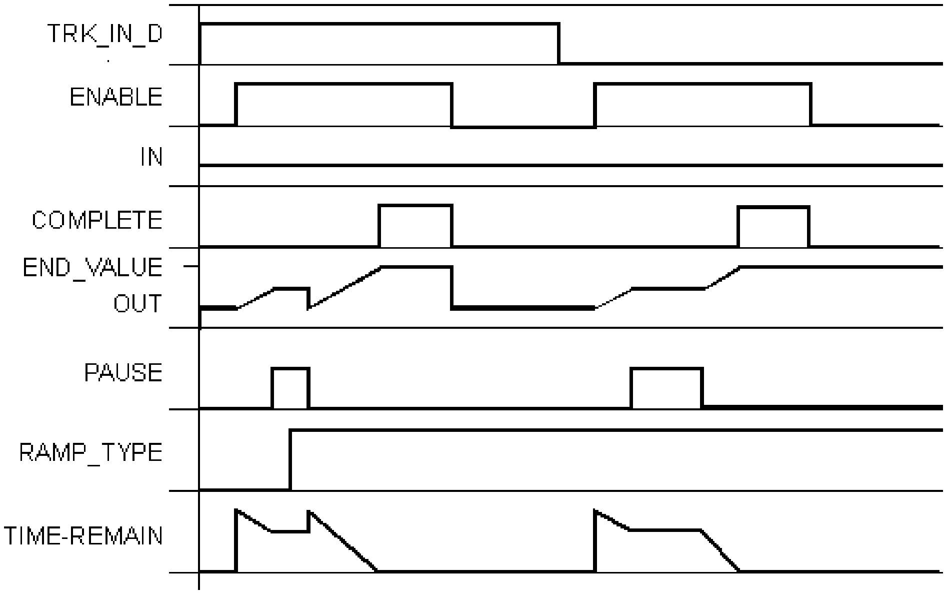
ENABLE	TRK_IN_D	PAUSE	COMPLETE	Ramp Starting Point	OUT
False	False	Forced False	False	N/A	Last OUT
True	False	False	True at OUT = END_VALUE	Last OUT	Ramp
False	True	Forced False	False	N/A	IN
True	True	False	True at OUT = END_VALUE	IN	Ramp
True	False	True	True at OUT = END_VALUE	N/A	Last OUT
True	True	True	True at OUT = END_VALUE	N/A	Last OUT

Output Tracking

You select output tracking by configuring the TRK_IN_D parameter. When TRK_IN_D is True, OUT tracks IN.

When the ENABLE input transitions from True to False:

- When the TRK_IN_D parameter is False, the OUT parameter is held at its last value.
- When TRK_IN_D is True, OUT tracks IN.



Ramp Function Block Output Tracking

Status Handling - Ramp Function Block

The output status is set to the worst status among the inputs.

Parameters - Ramp Function Block

The following table lists the system parameters for the Ramp function block:

Ramp Function Block System Parameters

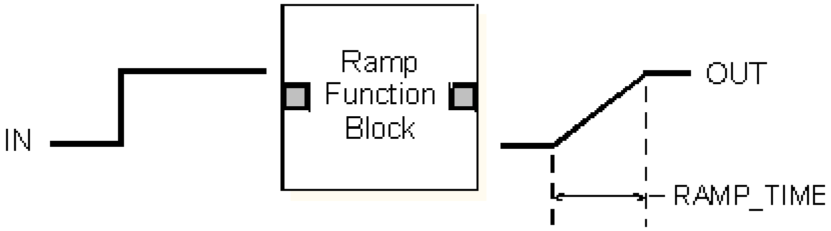
Parameter	Units	Description
COMPLETE	None	Indicates the ramp calculation is complete.
ENABLE	None	Discrete input value and status that activates the ramping calculation.
END_VALUE	EU of IN	The ramp endpoint input value and status.
IN	Determined by source	The analog input value and status that can be used as the ramp starting point.

Parameter	Units	Description
OUT	EU of IN	The analog output value and status.
PAUSE	None	Discrete input value and status that stops the ramping calculation. OUT and TIME_REMAIN are held at their last values.
RAMP_RATE	EU of IN per second	The increase or decrease rate of the ramp calculation.
RAMP_TIME	Seconds	The time period for the output to ramp from IN to END_VALUE.
RAMP_TYPE	None	The variable used for ramp calculation (False [0] = RAMP_TIME, True [1] = RAMP_RATE).
TIME_REMAIN	Seconds	The time left until COMPLETE transitions to TRUE.
TRK_IN_D	None	Initiates the output tracking function (True [1] = active, False [0] = inactive).

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Ramp Function Block

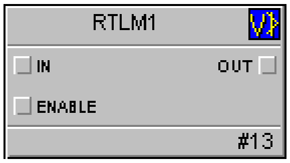
The Ramp function block is used to generate a ramping setpoint for a control loop. This allows you to change a setpoint smoothly to prevent process upset. You can use the Ramp function block to charge a tank, to change a process temperature slowly in increments, or to control a motor speed change. The following diagram shows an example of a ramping output from the Ramp function block:



Ramp Function Block Application Example

Rate Limit Function Block

The Rate Limit (RTLM) function block limits the rate of change of the output value to specified limits. The block supports signal status propagation. There are no modes or alarm detection in the Rate Limit Function Block.



Rate Limit Function Block

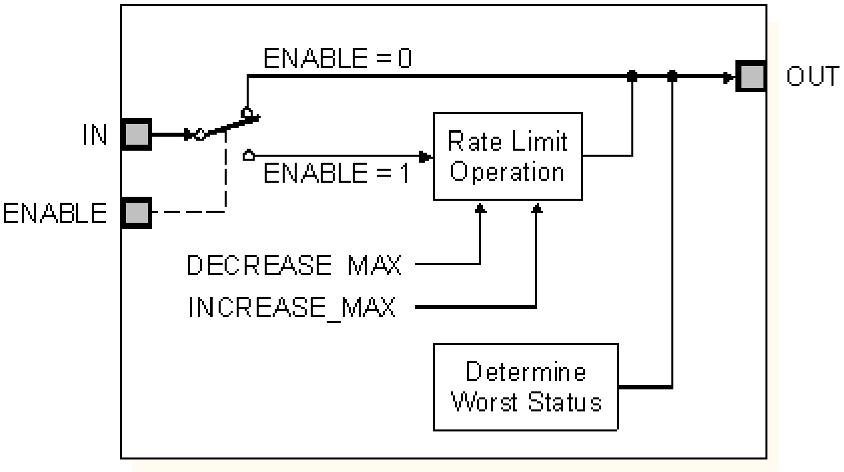
IN is the analog input value and status.

ENABLE activates rate limiting.

OUT is the analog output value and status.

Schematic Diagram - Rate Limit Function Block

The following diagram shows the internal components of the Rate Limit function block:



Rate Limit Function Block Schematic Diagram

Block Execution - Rate Limit Function Block

When the ENABLE input is True (1), rate limiting is performed on the input. When ENABLE is False (0), the output tracks the input.

The rate of change equation used in the Rate Limit function block is:

$$roc = \frac{IN - IN[t-1]}{\Delta t}$$

where:

roc = the calculated rate of change of the input between cycles

IN[t-1] = the value of IN from the previous cycle

t = the elapsed time since the previous cycle, in seconds.

If the rate of change is greater than the configured maximum increase value (INCREASE_MAX):

$$OUT = IN[t-1] + \Delta t * INCREASE\_MAX$$

If the rate of change is less than the configured maximum decrease value (DECREASE_MAX):

$$OUT = IN[t-1] + \Delta t * DECREASE\_MAX$$

If the rate of change is greater than DECREASE_MAX and less than INCREASE_MAX:

OUT = IN

Status Handling - Rate Limit Function Block

The output status is set to the worse status of IN and ENABLE.

Parameters - Rate Limit Function Block

The following table lists the system parameters for the Rate Limit function block:

Rate Limit Function Block System Parameters

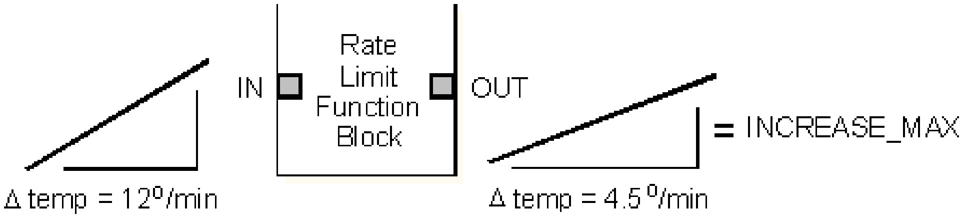
Parameter	Units	Description
DECR_LIMITED	None	When 1, indicates that OUT is not equal to IN due to a decreasing rate of change exceeding (being less than) DECREASE_MAX. Otherwise, the value is 0.
DECREASE_MAX	EU of IN per TIME_UNITS	The maximum decreasing rate of change allowed for OUT. DECREASE_MAX must be a negative number.
ENABLE	None	The discrete input that activates rate limiting.
IN	Determined by source	The analog input value and status.

Parameter	Units	Description
INCR_LIMITED	None	When 1, indicates that OUT is not equal to IN due to an increasing rate of change exceeding (being greater than) INCREASE_MAX. Otherwise, the value is 0.
INCREASE_MAX	EU of IN per TIME_UNITS	The maximum increasing rate of change allowed for OUT. INCREASE_MAX must be a positive number.
LIMITED	None	When 1, indicates that OUT is not equal to IN due to an excessive rate of change. Is the logical OR of DECR_LIMITED and INCR_LIMITED. Otherwise, the value is 0.
OUT	EU of IN	The analog output value and status.
ROC	EU of IN per TIME_UNITS	The filtered rate of change of the IN parameter. ROC is an indication that is independent of the scan-to-scan rate of change used by the block algorithm to determine OUT.
TIMECONST	Seconds	The time constant of the first order filter applied to the scan-to-scan rate of change calculation for the ROC parameter.
TIME_UNITS	Selectable time unit - seconds, minutes, hours, or days	Determines the time units for the ROC parameter. Also used to convert INCREASE_MAX and DECREASE_MAX to units of EU/sec.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Rate Limit Function Block

The Rate Limit function block keeps a controlled variable from changing too quickly. This can help keep the process stable. You can use the Rate Limit function block to keep a cook temperature from increasing too rapidly, as shown in the following diagram:



Rate Limit Function Block Application Example

In this example, you configure the INCREASE_MAX parameter as:

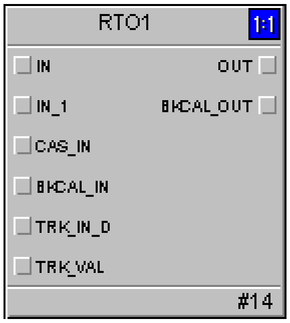
$$\frac{45\ \text{degrees}}{\text{min}} \times \frac{\text{min}}{60\ \text{sec}} = \frac{0.075\ \text{degrees}}{\text{sec}}$$

In addition, you can limit the decreasing rate of change to this rate by configuring the DECREASE_MAX parameter to -0.075 degrees/sec.

Ratio Function Block

The Ratio (RTO) function block is usually employed to drive the flow rate of a secondary flow stream to a setpoint that is a specified ratio of a primary flow stream. The block has two inputs: IN_1, which is the flow rate of the primary stream and IN, the flow rate of the secondary flow stream. The Ratio block's setpoint is the ratio (secondary flow rate)/(primary flow rate). This output is normally fed out to a PID flow controller for the secondary stream.

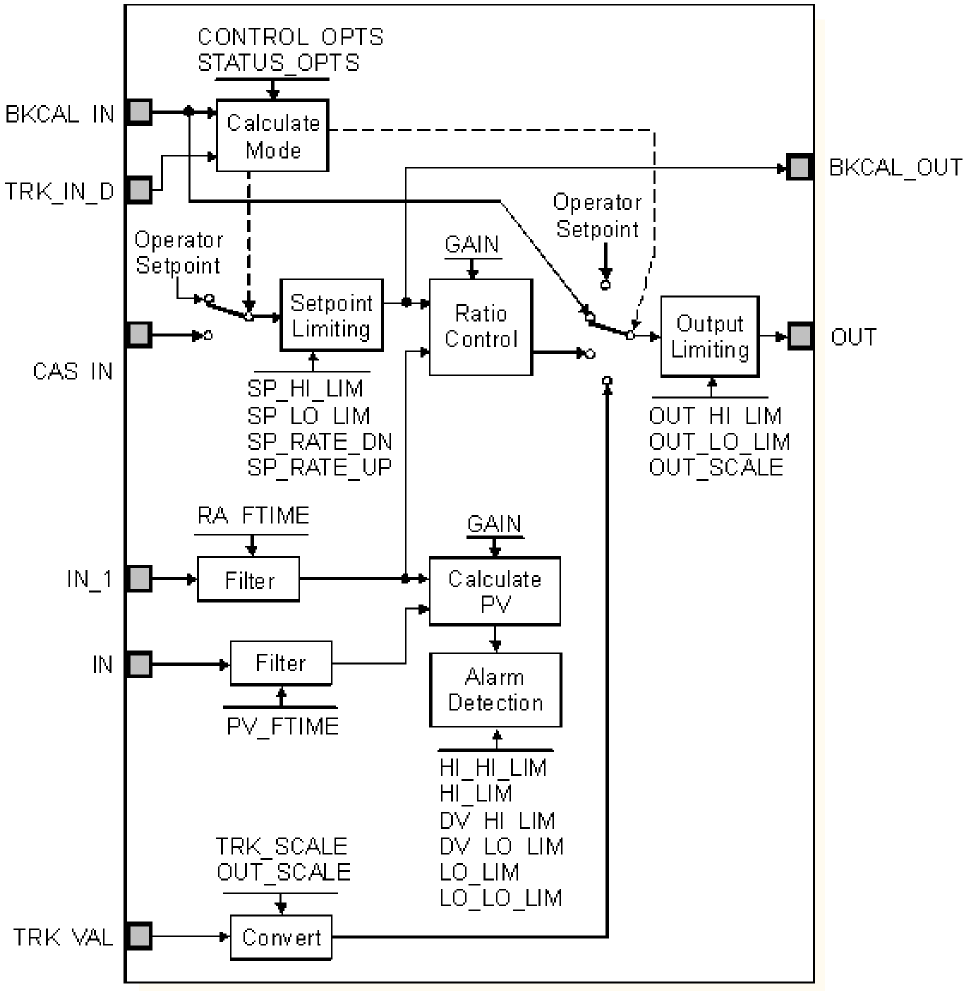
The Ratio function block supports signal filtering, mode control, output tracking, and alarm detection. This function block is normally used in a cascade configuration and supports normal block-to-block communication through the BKCAL channel.



Ratio Function Block

Schematic Diagram - Ratio Function Block

The following diagram shows the internal components of the Ratio function block:



Ratio Function Block Schematic Diagram

Block Execution - Ratio Function Block

In Cascade (Cas) or Automatic (Auto) mode, the Ratio function block computes the block output (OUT) from a ratio setpoint, an input, and a gain value:

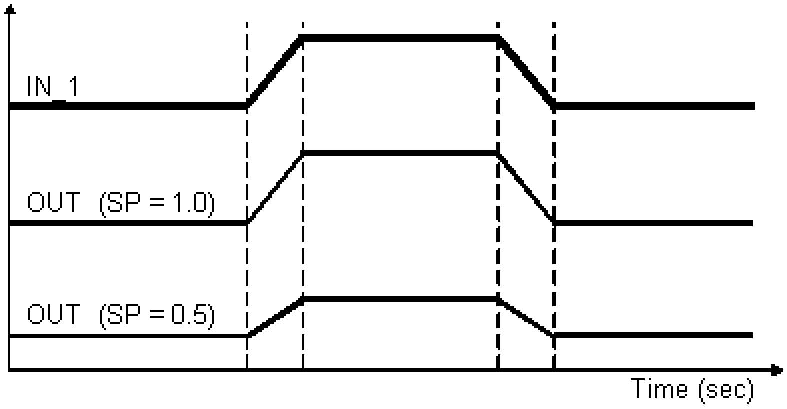
$$\mathrm{OUT} = \mathrm{SP} * \mathrm{IN}\ \_1 * \mathrm{GAIN}$$

where:

IN_1 is the input to which the ratio setpoint (SP) is applied after filtering with the RA_FTIME parameter value.

GAIN is the gain value used to range-adjust the result and control the decimal point location of the ratio.

The following diagram shows the timed response of the Ratio function block:



Ratio Function Block Timing Diagram

Select how the block executes by configuring input filters, tracking variables, setpoint limits, and output limits. The block's mode determines setpoint and output selection.

Input Filtering

Set the ratio filter time constant (RA_FTIME) to filter IN_1. Set the PV filter time constant (PV_FTIME) to filter IN. When IN is not connected, and BKCAL_IN and IN_1 are good, PV status is good.

Calculation of PV

In the Ratio function block, PV is the actual ratio. It is calculated as follows:

$$PV=\frac{IN}{IN\_1*GAIN}$$

The value of IN used in the calculation is the value of IN filtered by PV_FTIME if IN is wired (that is, its status is not BadNotConnected). If IN is not wired, the value used in the calculation is the value of BKCAL_IN if it is wired. Otherwise, the value used is that of OUT.

IN_1 is filtered by RA_FTIME before it is used in the calculation. PV is constrained within the EU range of PV_SCALE.

Tracking

You can specify output tracking with control options and parameters.

The Track Enable control option (CONTROL_OPTS) must be True (1) for the track function to operate. When the Track in Manual control option is True, tracking can be activated and maintained when the block is in Manual mode. When Track in Manual is False, the operator can override the tracking function when the block is in Man mode. Activating the track function causes the block's actual mode to go to Local Override (LO).

The tracking value parameter (TRK_VAL) specifies the value to be converted and tracked into the output when the track function is operating. The tracking scale parameter (TRK_SCALE) specifies the range of TRK_VAL.

When the track control parameter (TRK_IN_D) is True and the Track Enable control option is True, the TRK_VAL input is converted to the appropriate value and output in units of OUT_SCALE.

Setpoint and Output Limit Constraints

As part of download the output high and low limits are set to the configured values. If these limits have not been configured OUT_HI_LIM will be set to OUT_SCALE_HI and OUT_LO_LIM will be set to OUT_SCALE_LO. The following constraints apply to download or direct entry:

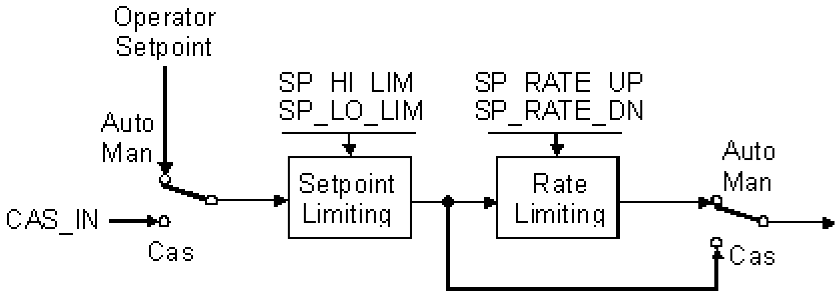
- SP_HI_LIM is restricted to PV_SCALE_HI+.1*(PV_SCALE_HI-PV_SCALE_LO).
- SP_LO_LIM is restricted to PV_SCALE_LO-.1*(PV_SCALE_HI-PV_SCALE_LO).
- OUT_HI_LIM is restricted to OUT_SCALE_HI+.1*(OUT_SCALE_HI-OUT_SCALE_LO).
- OUT_LO_LIM is restricted to OUT_SCALE_LO-.1*(OUT_SCALE_HI-OUT_SCALE_LO).

If the new scale causes a limit to be outside of these rules, the DeltaV system forces the limit within the rules.

The SP or OUT parameters are not changed as a result of changing the scale or limits. However, if OUT or SP are outside newly established limits, the values will be forced within the limits.

Setpoint Selection and Limiting

Setpoint selection is determined by the mode. The following diagram shows the method for setpoint selection:



Ratio Function Block Setpoint Selection

You can limit the setpoint by configuring the SP_HI_LIM and SP_LO_LIM parameters. In Cas mode, the setpoint comes from the CAS_IN input. In Auto mode, the setpoint is adjusted by the operator, and you can apply rate of change limits to the setpoint (SP_RATE_DN and SP_RATE_UP).

Output Selection and Limiting

Output selection is determined by mode. In Auto or Cas mode, the output is computed by ratio control. In Man mode, the output can be entered manually.

You can apply limits to the output by configuring the OUT_HI_LIM and OUT_LO_LIM parameters.

Bumpless Transfer on Mode Transitions

The BAL_TIME parameter determines bumpless transfer operations. When the mode transitions from a non-Auto mode to Auto mode or from any mode to Cas mode, an internal balancing bias is calculated that forces the block to initialize its output to a value that does not bump the output. The internal bias is ramped to zero over a period of time specified by BAL_TIME. When BAL_TIME is zero, the internal bias and ramp are not applied. When BAL_TIME is non-zero, the internal bias and ramp are applied.

When internal bias and ramp are not applied, the setpoint is back-calculated in all modes except Auto and Cas. However, there are cases when back calculation cannot be accomplished (such as when IN_1 is zero or when CAS_IN is connected to the output of a block that does not have initialization capability). Therefore, make sure the value in BAL_TIME provides a reasonable balancing time.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Out of Service – The block is in Out of Service (OOS) mode.

Local override – The block is in Local Override (LO) mode.

Block configuration error – IN_1 0 or GAIN = 0.

Modes - Ratio Function Block

The Ratio function block supports multiple modes:

Out of Service (OOS)

Initialization Manual (IMan)

Local Override (LO)

Manual (Man)

Automatic (Auto)

Cascade (Cas)

Remote Cascade (RCas)

You can configure the Man, Auto, Cas, and OOS modes as permitted modes for operator entry.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Alarm Detection - Ratio Function Block

Block alarm detection is based on the PV and SP values. You can configure the following alarm limits to compare to the PV value for alarm detection:

- High (HI_LIM)
- High high (HI_HI_LIM)
- Low (LO_LIM)
- Low low (LO_LO_LIM)

You can configure the following alarm limits to compare to the difference between the SP and PV values for deviation alarm detection:

- Deviation high (DV_HI_LIM) — The deviation of PV is above SP.
- Deviation low (DV_LO_LIM) — The deviation of PV is below SP.

IN must be connected to provide PV calculation and in order for the alarm limits to have meaning.

Status Handling - Ratio Function Block

The output status is based on the actual mode of the block and on the status of the IN_1 block input. When the IN_1 status is Bad, the block sheds to Man mode and the status of OUT becomes GoodCascade Const. In addition, you can choose the block to shed to Man mode when IN status is Bad by selecting the following status option (STATUS_OPTS):

Target to Manual if Bad IN

The status of IN is reflected in the PV value but has no effect on control operation.

Note You can set the status option in Manual or Out of Service mode only.

Parameters - Ratio Function Block

The following table lists the system parameters for the Ratio function block:

Ratio Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active).
ALARM_HYS	Percent of PV_SCALE	The amount the alarm value must return within the alarm limit before the associated active alarm condition clears. ALARM_HYS is limited to 50% of scale.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for dissipation of the internal balancing bias.
BKCAL_IN	EU of OUT_SCALE	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BKCAL_OUT	EU of PV_SCALE	The value and status required by the BKCAL_IN input of another block to prevent reset windup and provide bumpless transfer to closed loop control.

Parameter	Units	Description
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Ratio function block are Block configuration error, Local override, and Out of Service.
CAS_IN	EU of PV_SCALE	The remote setpoint value from another block.
CONTROL_OPTS	None	Control options. Allow you to specify control strategy options. The valid control options for the Ratio function block are No OUT Limits in Manual, Obey SP Limits if Cas or RCas, Use BKCAL_OUT With IN_1, Act on IR, Use PV for BKCAL_OUT, Track in Manual, Track Enable, SP Track Retained Target, SP-PV Track in LO or IMan, , SP-PV Track in Man.
DV_HI_ACT	None	The result of alarm detection associated with DV_HI_LIM. If DV_HI_ACT equals True, DV_HI_LIM has been exceeded.
DV_HI_LIM	EU of PV_SCALE	The amount by which PV can deviate above SP before the deviation high alarm is triggered. When this limit is exceeded, DV_HI_ACT is set to True. The deviation of PV is above SP.
DV_LO_ACT	None	The result of alarm detection associated with DV_LO_LIM. If DV_LO_ACT equals True, DV_LO_LIM has been exceeded.
DV_LO_LIM	EU of PV_SCALE	The amount by which PV can deviate below SP before the deviation low alarm is triggered. When this limit is exceeded, DV_LO_ACT is set to True. Note that DEV_LO_LIM is a negative number and is compared against (PV - SP).
FRSIRB_OPTS	None	FRSI add-on I/O option. Supported option is Treat_IN_1 as wild.
GAIN	None	The multiplier used to scale the ratio value.
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.
HI_HI_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the high high alarm condition.
HI_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the high alarm condition.
IN	Defined by supplying block	The analog input value and status used with IN_1 to calculate PV.

Parameter	Units	Description
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.
IN_1	Defined by supplying block	The analog input value and status used to calculate OUT.
LO_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the low low alarm condition.
MODE	None	Parameter used to request, show, and set the block operating state.
OUT	EU of OUT_SCALE	The analog output value and status.
OUT_HI_LIM	EU of OUT_SCALE	The maximum output value allowed.
OUT_LO_LIM	EU of OUT_SCALE	The minimum output value allowed.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
PV	EU of PV_SCALE	The process variable used in block execution and alarm limit detection.
PV_FTIME	Seconds	The time constant of the first-order PV filter applied to IN. It is the time required for a 63% change in IN.
PV_SCALE	None	The high and low scale values, the engineering units code, and the number of digits to the right of the decimal point associated with PV.
RA_FTIME	Seconds	The time constant of the first-order ratio filter applied to IN_1. It is the time required for a 63% change in IN_1.
RCAS_IN	EU of SP_SCALE	The remote analog setpoint value and status. Input provided by a device or the output of another block.
RCAS_OUT	EU of PV_SCALE	The output provided for bumpless mode transfer and reset limiting by the source of RCAS_IN. Essentially the equivalent of BKCAL_OUT for RCAS_IN.
SHED_OPT	None	Defines action to be taken on remote control device time out.*
SHED_TIME	Seconds	The maximum allowable time between RCAS_IN and ROUT_IN updates. If exceeded, mode shedding takes place.

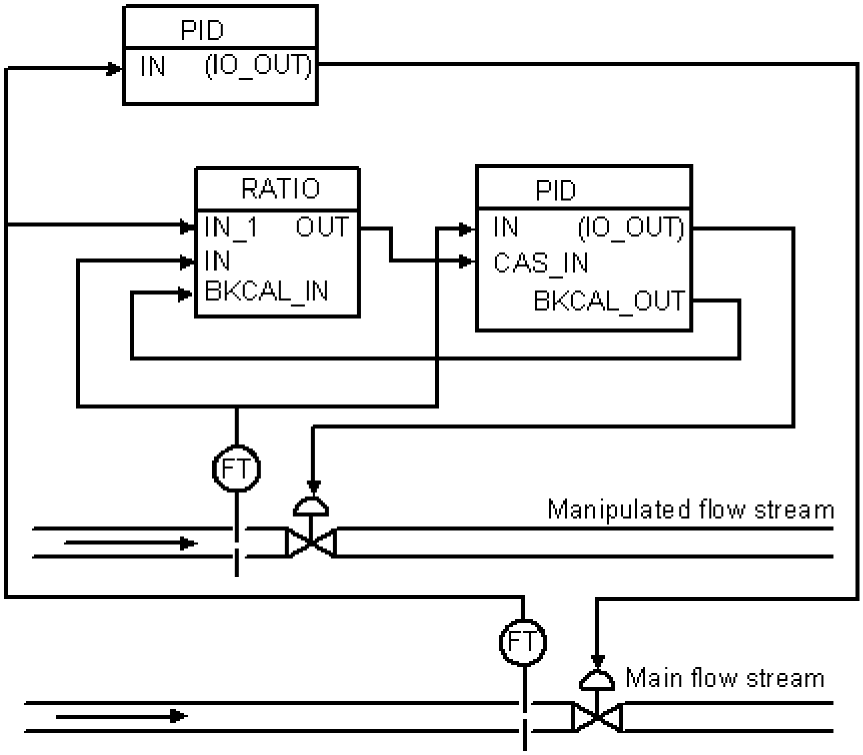
Parameter	Units	Description
SP	EU of PV_SCALE	The block's setpoint value.
SP_HI_LIM	EU of PV_SCALE	The highest setpoint value allowed.
SP_LO_LIM	EU of PV_SCALE	The lowest setpoint value allowed.
SP_RATE_DN	EU of PV_SCALE per second	Ramp rate at which downward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_RATE_UP	EU of PV_SCALE per second	Ramp rate at which upward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_WRK	EU of PV_SCALE	The working setpoint of the block. It is the result of setpoint limiting and setpoint rate of change limiting.
STDEV_CAP	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.
STDEV	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The standard deviation of PV. For analog control blocks in AUTO, mean is assumed to be the SP. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The supported status option for the Ratio function block is Target to Manual if Bad IN.
TRK_IN_D	None	Discrete input that initiates the external tracking function.
TRK_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the external tracking value (TRK_VAL).
TRK_VAL	EU of TRK_SCALE	The analog input used in the external tracking function.

Note Default values and datatype information for the parameters are available by expanding the Parameter View window.

Application Information - Ratio Function Block

Basic Setup

The following diagram shows the basic setup of the Ratio function block:



Basic Setup for Ratio Flow Control

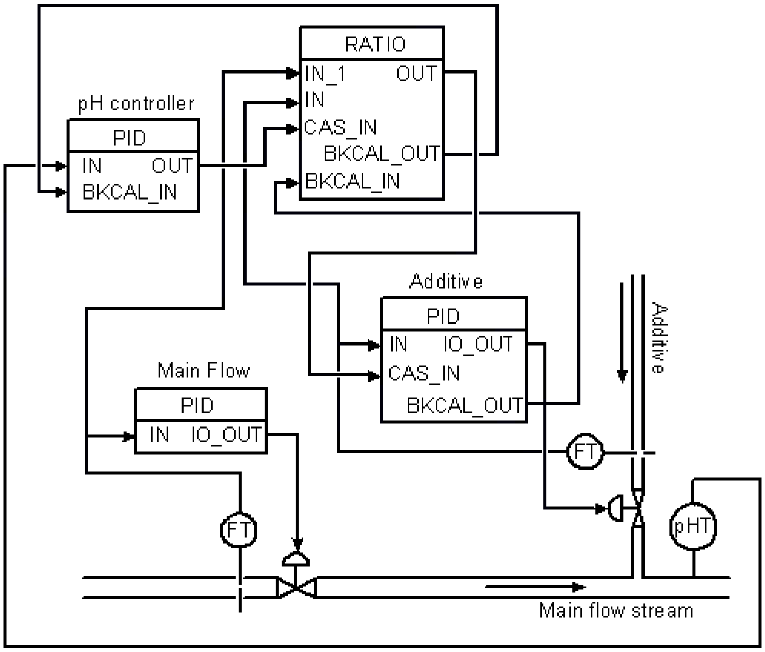
In this example, there are two flow streams: the primary (main) flow stream and the secondary (manipulated) flow stream. The primary stream's flow rate is set and governed by a conventional PID loop. The Ratio function block and its slave PID make the manipulated flow stream a specified fraction of the primary flow. The primary stream's flow rate is sent to the Ratio function block where the ratio setpoint is used to determine the desired flow rate of the manipulated stream. The output of the Ratio block is then cascaded into a slave flow controller for the secondary stream, where it is used as a setpoint for this stream.

Note The Ratio function block does not actually require the secondary (manipulated) flow rate. The input IN is optional. If this input is provided, it is used to calculate the actual ratio. This ratio is stored as the Ratio block's PV, which can be used for alarming.

In the above figure, it is implied that the Ratio function block runs in Automatic mode with the desired ratio set manually by an operator.

pH Control

The pH of a stream can be controlled by adding caustic or acid to a primary flow stream. The additive must be added in the proper ratio to get the desired pH. The setup in the following figure shows how this can be done using the Ratio function block.



pH Control

Scaler Function Block

The Scaler (SCLR) function block provides scaling and dimensional consistency between two values of different engineering units. The block converts the input value to the specified scale and generates an output value.

The Scaler function block supports signal status propagation. There are no modes or alarm detection in the block.

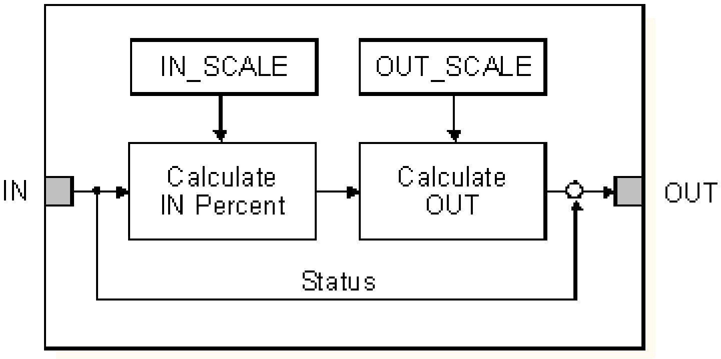
Scaler Function Block

IN is the analog input value and status.

OUT is the analog output value and status.

Schematic Diagram - Scaler Function Block

The following diagram shows the internal components of the Scaler function block:



Scaler Function Block Schematic Diagram

Block Execution - Scaler Function Block

The Scaler function block uses the input value, input range, and output range to compute the scaled output value. The percentage value of the input in its configured range is the same as the percentage value of the output in its configured range.

The block converts the input value (IN) to a percentage of the specified IN_SCALE, and next converts this percentage value back into engineering units as specified by OUT_SCALE to generate the block output (OUT). The following equations are used:

$$\text{IN Percent} = \text{IN\_SCALE.Low} + \frac{\text{IN} - \text{IN\_SCALE.Low}}{(\text{IN\_SCALE.High} - \text{IN\_SCALE.Low})}$$

$$\text{OUT} = \text{OUT\_SCALE.Low} + \text{IN Percent} \times (\text{OUT\_SCALE.High} - \text{OUT\_SCALE.Low})$$

Note The Low and High extensions in this example stand for the low and high value fields of the IN_SCALE and OUT_SCALE parameters.

Status Handling - Scaler Function Block

The output status is set to the input status.

Parameters - Scaler Function Block

The following table lists the system parameters for the Scaler function block:

Scaler Function Block System Parameters

Parameter	Units	Description
IN	Determined by source	The analog input value and status.
IN_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with IN.
OUT	EU of OUT_SCALE	The analog output value and status.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

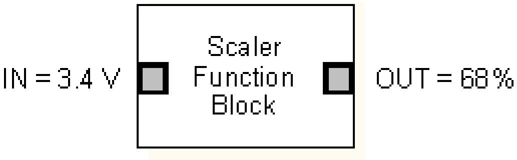
Application Information - Scaler Function Block

You can use the Scaler function block to convert units for calculations that are performed in other function blocks. The following example shows how you can use the Scaler function to convert a 0 to 5 volt signal to 0 to 100% of scale.

Scaler Function Block Configuration Settings Example

Parameter	Low Value	High Value
IN_SCALE	0.0	5.0
OUT_SCALE	0.0	100.0

The following figure is an example function block diagram for this application:



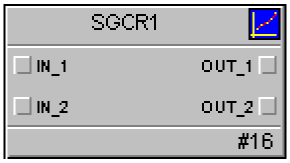
Scaler Function Block Diagram Example

Signal Characterizer Function Block

The Signal Characterizer (SGCR) function block characterizes or approximates any function that defines an input/output relationship. The function is defined by configuring as many as twenty X,Y coordinates. The block interpolates an output value for a given input value using the curve defined by the configured coordinates. Two separate analog input signals can be processed simultaneously to give two corresponding separate output values using the same defined curve.

The Signal Characterizer function block supports signal status propagation. There are no standard alarms in this function block. Custom alarms are supported.

The Signal Characterizer function block correlates the input IN_1 to the output OUT_1 and the input IN_2 to the output OUT_2 according to the configured curve. You configure the curve by defining as many as twenty pairs of X,Y values in the CURVE_X and CURVE_Y parameters. The CURVE_X array defines the input values (X1 to X20) and the CURVE_Y array defines the output values (Y1 to Y20).



Signal Characterizer Function Block

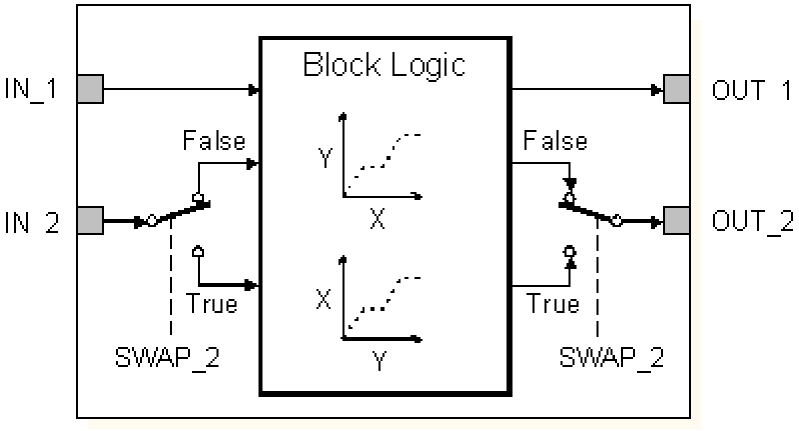
IN_1 and IN_2 are the input values to the block.

OUT_1 is the output associated with IN_1.

OUT_2 is the output associated with IN_2.

Schematic Diagram - Signal Characterizer Function Block

The following diagram shows the internal components of the Signal Characterizer function block:



Signal Characterizer Function Block Schematic Diagram

Block Execution - Signal Characterizer Function Block

For any given input value, the Signal Characterizer function block determines where the input lies in CURVE_X and calculates the slope of that segment using the point-slope method:

$y = mx + b$

where

m = slope of the line

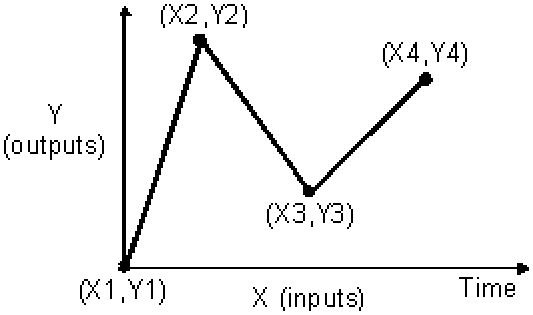
b = y-intercept of the line.

Using this formula, the block derives an output value that corresponds to the input. When the input lies beyond the range configured in CURVE_X, the output is clamped to the corresponding limit in the CURVE_Y array.

CURVE_X

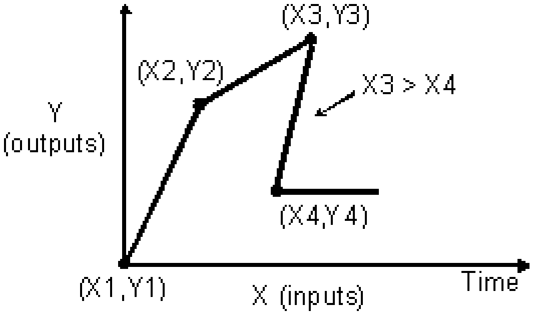
The CURVE_X values must be defined in ascending order. The X1 element must be the smallest value, and each following X value must be greater than the previous X value. When the X values are not configured in ascending order, a block configuration error is set and the last X value that is greater than or equal to the previous one is used as the curve endpoint.

The following diagram shows an example of a valid X value configuration:



Signal Characterizer Function Block Valid CURVE_X Values Example

The following diagram shows an example of an invalid X value configuration:



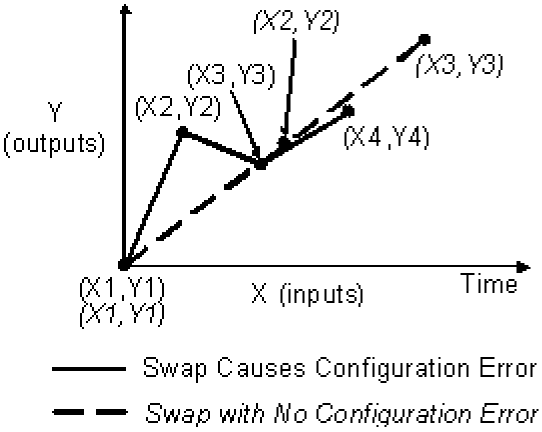
Signal Characterizer Function Block Invalid CURVE_X Values Example

This curve has an invalid definition because X3 is greater than X4. Between these points, the Y value is undefined because it can be any value from Y3 to Y4. In this configuration, the X3,Y3 pair becomes the endpoint for the curve definition. To use the X4,Y4 pair, you must designate X4 to be greater than or equal to X3.

SWAP_2

The SWAP_2 parameter swaps the X and Y axes used for OUT_2. When the SWAP_2 parameter is True, IN_2 references the CURVE_Y values and OUT_2 references the CURVE_X values. In addition, the IN_2 units change to Y_UNITS and the OUT_2 units change to X_UNITS.

The block sets a configuration error when SWAP_2 is True and the CURVE_Y elements are not configured in an increasing manner. The following example shows how the block configuration error (BLOCK_ERR) is set during a SWAP_2 action:



Signal Characterizer Function Block SWAP_2 Configuration Error Example

When swap is in effect, the first curve has an invalid definition because Y3 is less than Y2. In this configuration, the X2,Y2 pair becomes the endpoint for the swapped curve definition when processing IN_2. Note that the X4,Y4 pair is the valid endpoint when processing IN_1.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Out of Service – The block is in Out of Service (OOS) mode.

Block configuration error – Set if an invalid definition occurs in less than the first 20 points and the X value of the X,Y pairs beyond the valid definition are non-zero.

Modes - Signal Characterizer Function Block

The Signal Characterizer function block supports two modes:

Out of Service (OOS)

Automatic (Auto)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Signal Characterizer Function Block

The OUT_1 status is set to the IN_1 status and the OUT_2 status is set to the IN_2 status. When one of the curve limits is reached, the appropriate limit is set in the substatus.

Parameters - Signal Characterizer Function Block

The following table lists the system parameters for the Signal Characterizer function block:

Signal Characterizer Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Signal Characterizer function block are Block configuration error and Out of Service.
BYPASS	None	When BYPASS is enabled in the Signal Characterizer function block, it bypasses the signal characterization. OUT_1 equals IN_1 and OUT_2 equals IN_2. To enable BYPASS, set the BYPASS parameter to On and select the CONTROL_OPTS Bypass Enable option.
CONTROL_OPTS	None	Control options. Allow you to specify control strategy options. The supported control option for the Signal Characterizer function block is Bypass Enable.
CURVE_X	Determined by source	The array of input curve values (as many as 20 points).
CURVE_Y	Determined by source	The array of output curve values (as many as 20 points).
IN_1	Determined by source	The first analog input value and status.
IN_2	Determined by source	The second analog input value and status.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.

Parameter	Units	Description
OUT_1	EU of IN	The analog output value and status related to IN_1.
OUT_2	EU of IN	The analog output value and status related to IN_2.
SWAP_2	None	Swaps the X and Y axes used for OUT_2.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

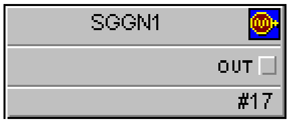
Application Information - Signal Characterizer Function Block

You can use the Signal Characterizer function block as a curve fitting function. For example, you can scale a 4 to 20 mA input signal to a 0 to 100% output value using the block. You can also use the block to convert measurements from a split-range or other nonlinear device or from a dual-temperature control device used for both heating and cooling.

You can use the Signal Characterizer function block to alter the relationship of a PID function block output to valve position and gain more linear control over a critical region.

Signal Generator Function Block

The Signal Generator (SGGN) function block produces an output signal that simulates a process signal. The block uses a specified combination of a sine wave, a square wave, a bias value, and a random value to generate the output signal. There are no modes or alarm detection in the Signal Generator function block.

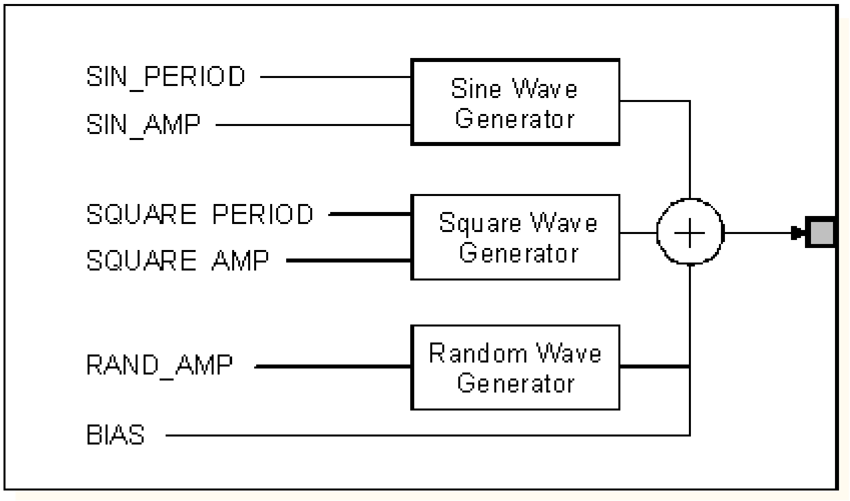


Signal Generator Function Block

OUT is the output value and status.

Schematic Diagram - Signal Generator Function Block

The following figure shows the internal components of the Signal Generator function block:



Signal Generator Function Block Schematic Diagram

Block Execution - Signal Generator Function Block

The Signal Generator function block generates the sine wave, square wave, and random signal components that you specify. The block also applies the specified bias value and generates the output value. The output status is set to Good: Non-cascade.

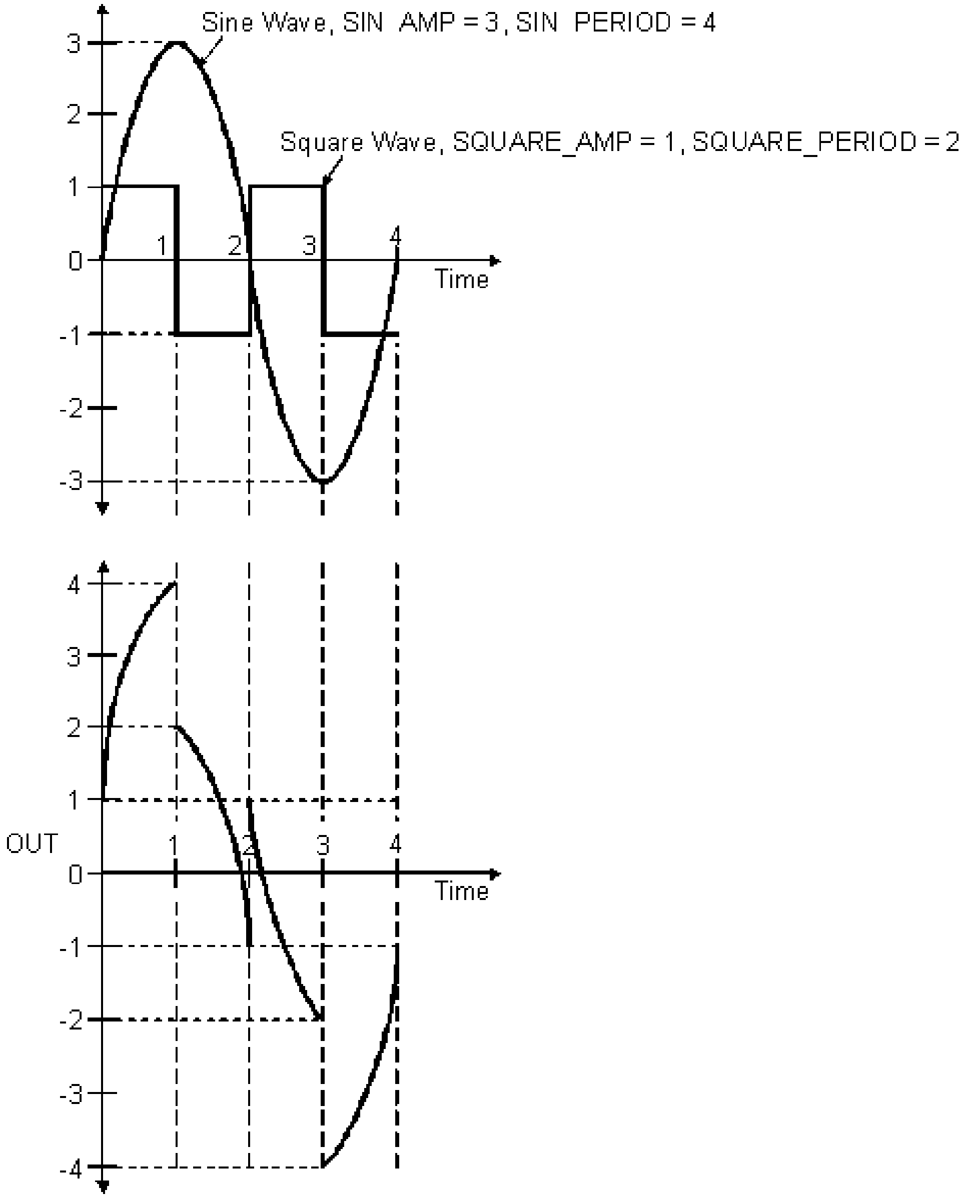
You specify the sine wave period parameter (SIN_PERIOD) and the sine wave amplitude parameter (SIN_AMP) to generate the sine wave component.

You specify the square wave period parameter (SQUARE_PERIOD) and the square wave amplitude parameter (SQUARE_AMP) to generate the square wave component.

You specify the random signal amplitude parameter (RAND_AMP) and the first-order filter (RAND_FTIME) to generate the random signal component.

You specify the bias parameter (BIAS) to be applied to the signal.

The following figure shows an example of the timed response of the Signal Generator function block:



Signal Generator Function Block Timing Diagram

Status Handling - Signal Generator Function Block

The output status is always Good: Non-cascade.

Parameters - Signal Generator Function Block

The following table lists the system parameters for the Signal Generator function block:

Signal Generator Function Block System Parameters

Parameter	Units	Description
BIAS	None	The bias value.
OUT	None	The output value and status.
RAND_AMP	None	The maximum amplitude of the generated random component.
RAND_FTIME	Seconds	The filter time constant for the random signal component. It is the time required for a 63% change in the random signal.
SIN_AMP	None	The amplitude of the generated sine wave component.
SIN_PERIOD	Seconds	The period of the generated sine wave component. This parameter determines the frequency of the sine wave.
SQUARE_AMP	None	The amplitude of the generated square wave component.
SQUARE_PERIOD	Seconds	The period of the generated square wave component. This parameter determines the frequency of the square wave.

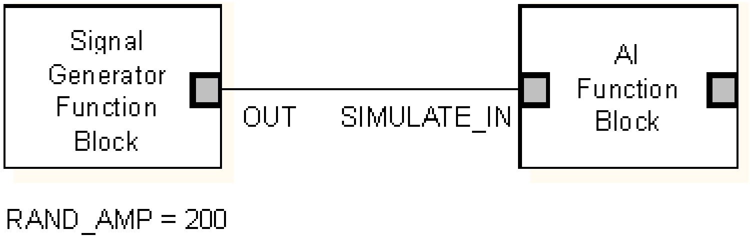
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Signal Generator Function Block

You can use the Signal Generator function block to simulate a signal and to send the generated output to other blocks to test a control strategy. This block is also useful for simulating process disturbances for loop testing.

Application Example: Testing

You can use the Signal Generator function block to perform configuration tests or to test alarms and user interfaces before startup. For example, when you want to simulate a temperature input with a range of -200 to 200°F, you configure the RAND_AMP parameter to 200. Next, you connect the Signal Generator block output signal (OUT) to an Analog Input (AI) function block SIMULATE_IN input, as in the following figure:

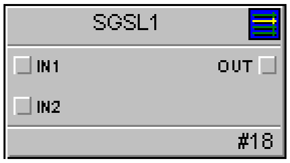


Signal Generator Function Block Application Example

The Signal Generator function block generates a random value between -200 and 200. You can use these values to test how your control strategy responds to all of the possible input values. In addition, you can test DeltaV Operate by seeing what happens when the generated signal to the AI function block exceeds a limit, triggers an alarm, and/or notifies the operator.

Signal Selector Function Block

The Signal Selector (SGSL) function block selects the maximum, minimum, or average of as many as sixteen input values and places it at the output. The block supports signal status propagation. There are no modes or alarm detection in the Signal Selector function block.



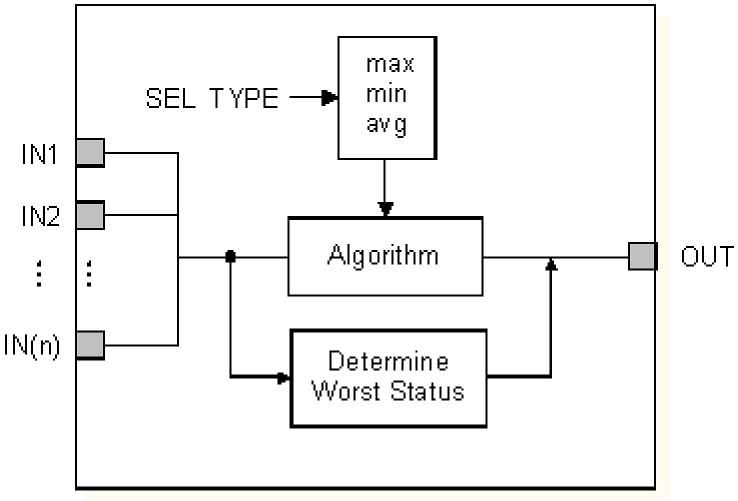
Signal Selector Function Block

IN1 through IN[n] are the analog input values and statuses (as many as 16 inputs).

OUT is the selected analog value and status.

Schematic Diagram - Signal Selector Function Block

The following figure shows the internal components of the Signal Selector function block:



Signal Selector Function Block Schematic Diagram

Block Execution - Signal Selector Function Block

The Signal Selector function block reads the values and statuses of as many as sixteen inputs. The number of inputs is an extensible parameter. The block default is two inputs. You add inputs by selecting the function block diagram, clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs. This creates additional input connectors for the block.

You configure the selector type parameter (SEL_TYPE) to specify which of the three available methods (algorithms) is used to select the output:

- SEL_TYPE = max selects the maximum value of the inputs.
- SEL_TYPE = min selects the minimum value of the inputs.
- SEL_TYPE = avg calculates the average value of the inputs.

Status Handling - Signal Selector Function Block

The output status is set to the worst status among the inputs.

If the input status is BAD, the value of that input is ignored. If all of the inputs have a BAD status, the block executes using the inputs with BAD status.

Parameters - Signal Selector Function Block

The following table lists the system parameters for the Signal Selector function block:

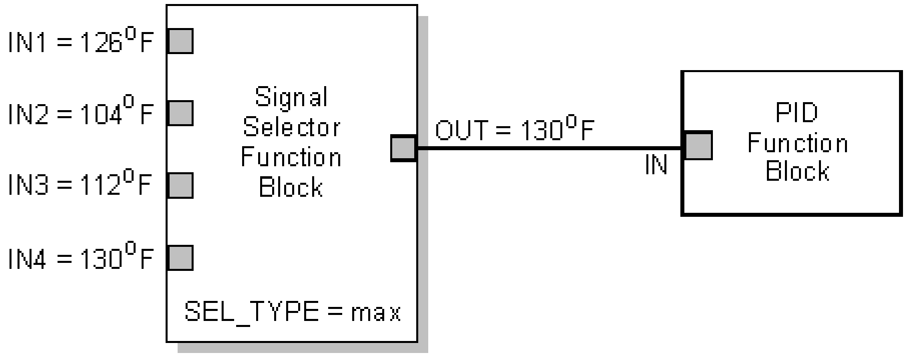
Signal Selector Function Block System Parameters

Parameter	Units	Description
IN1 to IN16	Determined by source	The analog input values and statuses. The number of inputs is an extensible parameter.
OUT	EU of IN	The analog output value and status.
SEL_TYPE	None	Specifies the selection method (max = maximum IN value, min = minimum IN value, avg = average IN value).

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

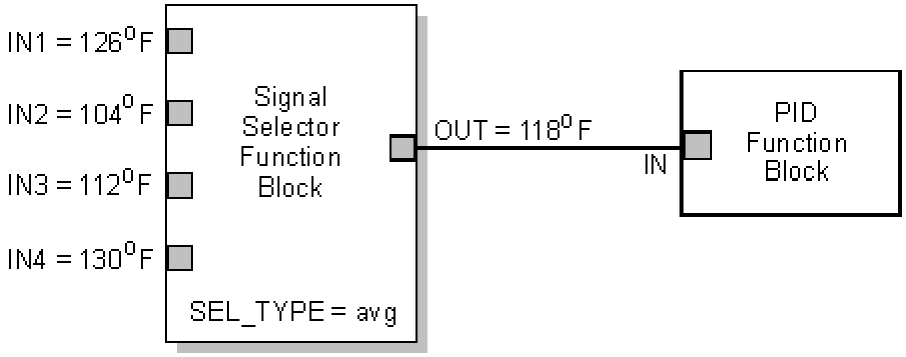
Application Information - Signal Selector Function Block

You can use the Signal Selector function block to select the maximum temperature input from four inputs and send it to a PID function block to control a process water chiller, as in the following figure:



Signal Selector Function Block Application Example (SEL_TYPE = max)

Or, you can use the block to calculate the average temperature of the four inputs, as in the following figure:



Signal Selector Function Block Application Example (SEL_TYPE = avg)

Splitter Function Block

The Splitter (SPLTR) function block takes a single input and calculates two outputs based on specified coordinate values. This allows an integrating controller to drive two outputs without winding up when either or both outputs are constrained.

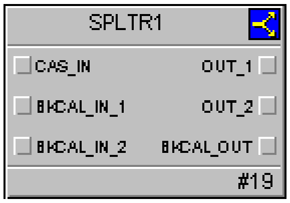
The Splitter function block supports mode control and signal status propagation. There are no standard alarms in this function block. Custom alarms are supported.

The transfer function for each output is a straight slope described by its endpoints. The control regions defined by the slopes can be separate or can overlap, but the low input limit is determined by the first output (OUT_1) and the high input limit is determined by the second output (OUT_2).

The block's normal mode is Cascade (Cas). You can isolate the block for testing by using Automatic (Auto) mode and adjusting the setpoint. Manual (Man) is not a permitted mode.

When a block attached to an output requests initialization, one of the following actions might occur:

- When the other output is not in Cas mode, the block attached to the input is initialized.
- When the other output is in Cas mode, this output returns to the value calculated from its slope in a specified time period.



Splitter Function Block

CAS_IN is the remote setpoint from another block.

BKCAL_IN_1 is the value and status reflecting the BKCAL_OUT of the lower block associated with OUT_1. It is used for initialization and to prevent windup in upstream blocks.

BKCAL_IN_2 is the value and status reflecting the BKCAL_OUT of the lower block associated with OUT_2. It is used for initialization and to prevent windup in upstream blocks.

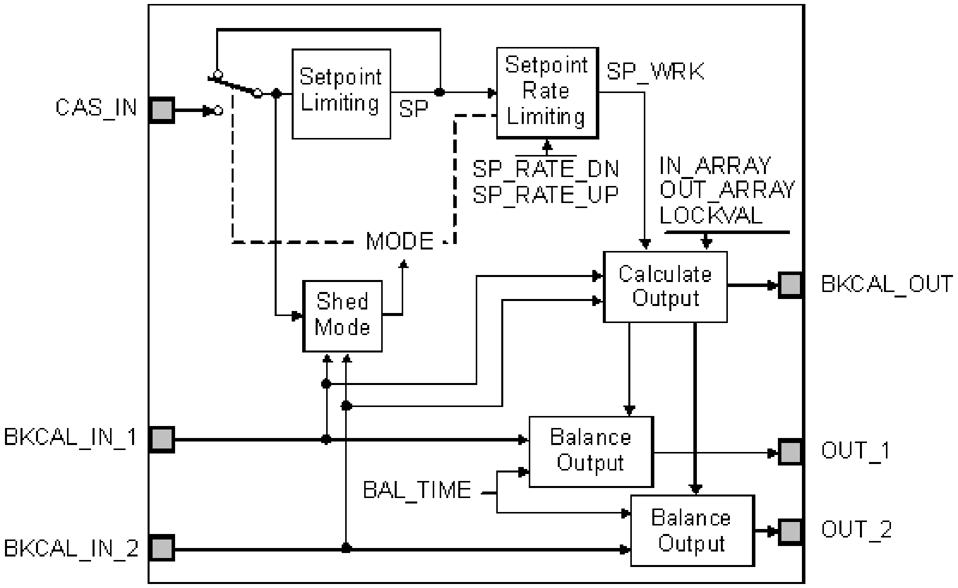
OUT_1 is the first output value and status.

OUT_2 is the second output value and status.

BKCAL_OUT is the value and status required by the BKCAL_IN input of the upstream block to prevent reset windup and to provide bumpless transfer to closed loop control.

Schematic Diagram - Splitter Function Block

The following figure shows the internal components of the Splitter function block:



Splitter Function Block Schematic Diagram

Block Execution - Splitter Function Block

You select how the outputs are calculated with parameter configuration. The inputs wired to the block determine the outputs.

Calculating Outputs

The block outputs (OUT_1 and OUT_2) are calculated from the block setpoint (SP) using the slope and limits established by the IN_ARRAY and OUT_ARRAY parameters. The four values of the IN_ARRAY determine the SP range used to calculate outputs based on the four values of the OUT_ARRAY output range. The following table illustrates the relationship between the block outputs and the array elements:

Splitter Function Block Output Assignments

Block Output	IN_ARRAY Element		OUT_ARRAY Element	
	Starting SP Value	Ending SP Value	Output Value for Starting SP Value	Output Value for Ending SP Value
OUT_1	X11	X12	Y11	Y12
OUT_2	X21	X22	Y21	Y22

Some constraints are enforced by the controller to guarantee useful output values:

- X12 must be greater than X11
- X22 must be greater than X21
- X21 must be greater than or equal to X11

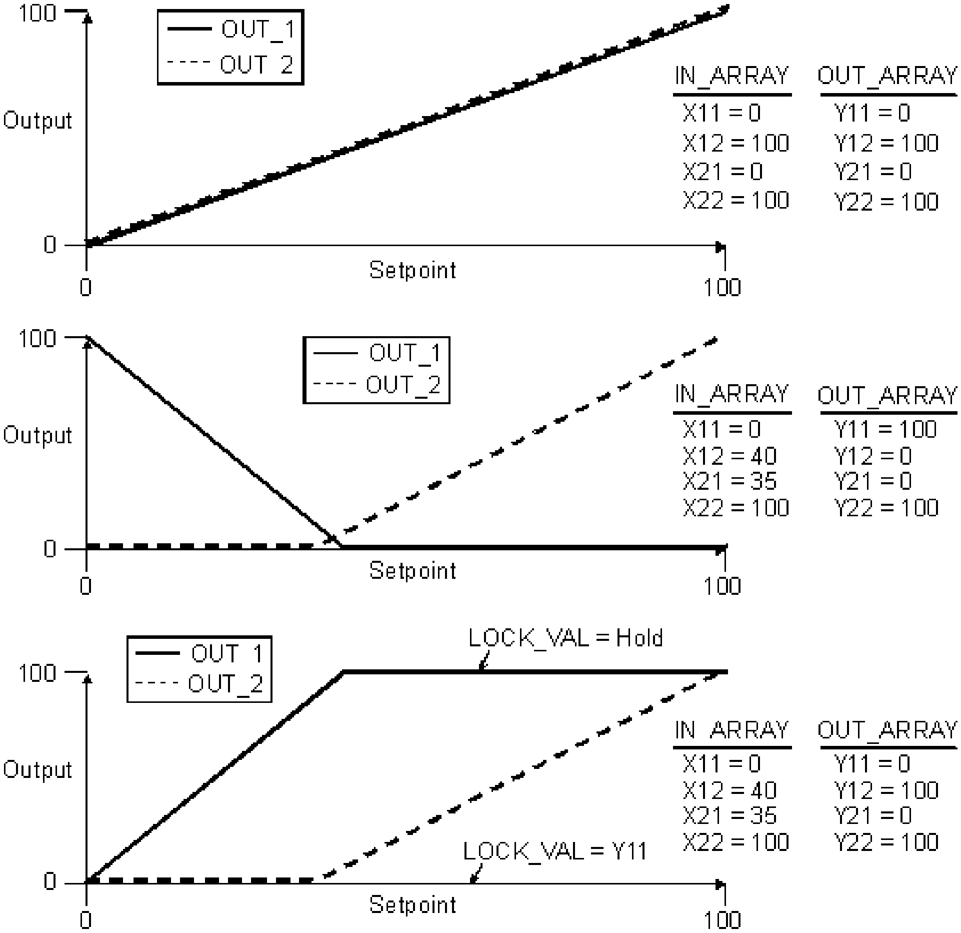
Violation of any of these constraints locks the block into OOS mode and sets the BLOCK_ERR parameter to Configuration Error.

The OUT_1 parameter is set to Y11 when the SP value exceeds X12 when the LOCKVAL parameter is defined as follows:

```
OUT_1 is Y11 when SP > X12
```

Hysteresis equal to five percent of the X11-to-X12 span is used to prevent OUT_1 from jumping between Y11 and Y12 at X12.

The following figure shows the output values for three examples of IN_ARRAY and OUT_ARRAY elements:



Splitter Function Block Execution Example

Setpoint Limiting

The setpoint is limited to the endpoints of the IN_ARRAY parameter values.

In Auto mode, you can apply rate of change limits to the setpoint by configuring the SP_RATE_UP and SP_RATE_DN parameters.

Modes - Splitter Function Block

The Splitter function block supports the following modes:

Out of Service (OOS)

Initialization Manual (IMan)

Automatic (Auto)

Cascade (Cas)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Splitter Function Block

The statuses of OUT_1 and OUT_2 are determined by the statuses of BKCAL_IN_1 and BKCAL_IN_2 and the actual mode of the block.

When a BKCAL_IN input sees that its downstream block is not in Cas mode, the Splitter function block sets the corresponding OUT value to the BKCAL_IN value. However, this may not be the same value that is calculated by the splitter algorithm. When the mode of the downstream block is changed to Cas, the difference between the calculated output and the back-calculation input is computed and the difference is added to the calculated output. Next, the difference is reduced to zero over the time defined by the BAL_TIME parameter. Choose a BAL_TIME value of at least two times the RESET value in the upstream PID block to minimize process disruptions.

When both BKCAL_IN_1 and BKCAL_IN_2 indicate that the downstream blocks are not in Cas mode or have Bad status, the first downstream block that goes to Cas mode causes the upstream block to initialize so that there is no difference between the calculated output and the back-calculation input. This provides bumpless transfer for the first downstream block.

Limit Handling

The Splitter function block is designed to combine the limit information from the two downstream blocks into limits for the upstream block. The general principle is to allow the upstream block to continue control for as long as possible.

The upstream block is high-limited (BKCAL_OUT of the Splitter block has high-limited status) when one of the following statements is true:

- Both downstream blocks are high-limited (both BKCAL_INS of the Splitter block have high-limited status).

  or

- One downstream block is high-limited, the associated output slope is positive, and the other block has Bad status or is not in Cas mode.

  or

- One downstream block is low-limited, the associated output slope is negative, and the other block has Bad status or is not in Cas mode.

  or

- SP is greater than or equal to X22.

The upstream block is low-limited when one of the following statements is true:

- Both downstream blocks are low-limited.

  or

- One downstream block is low-limited, the associated output slope is positive, and the other block has Bad status or is not in Cas mode.

  or

- One downstream block is high-limited, the associated output slope is negative, and the other block has Bad status or is not in Cas mode.

  or

- SP is less than or equal to X11.

Parameters - Splitter Function Block

The following table lists the system parameters for the Splitter function block:

Splitter Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for an output to match its calculated value.

Parameter	Units	Description
BKCAL_IN_1	Units of SP of block connected to OUT_1	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BKCAL_IN_2	Units of SP of block connected to OUT_2	The value and status reflecting the BKCAL_OUT of the lower block associated with OUT_2. This information is used for initialization and to prevent windup in upstream blocks.
BKCAL_OUT	EU of SP	The value and status required by the BKCAL_IN input of an upstream block that is used to prevent reset windup and to provide bumpless transfer to closed loop control.
BLOCK_ERR	None	The summary of active error conditions associated with the block. Block errors for the Splitter function block are Block configuration error and Out of Service.
CAS_IN	EU of SP	The remote setpoint value from an upstream block.
HYSTVAL	Percent of IN_ARRAY[2][1] - IN_ARRAY[2][1]	The hysteresis value used in Splitter to determine whether to set the target at the locked value.
IN_ARRAY	None	Array of four entries (X11, X12, X21, X22) that defines the setpoint range starting and ending points used in the calculation of OUT_1 (X11 and X12) and OUT_2 (X21 and X22). X11, X12, X21, and X22 are standard industry terms that correlate to the DeltaV parameters IN_ARRAY[1][1], IN_ARRAY[2][1], IN_ARRAY[3][1], and IN_ARRAY[4][1], respectively.
LOCKVAL	None	Specifies OUT_1 to hold or to be driven to OUT_ARRAY[1][1] when the SP exceeds the upper end of the range defined by IN_ARRAY[2][1].
MODE	None	Parameter used to request and show the source of the setpoint used by the block.
OUT_1	Units of block connected to OUT_1	The primary output value and status calculated by the block based on SP (in Auto and Cas modes).
OUT_2	Units of block connected to OUT_2	The primary output value and status calculated by the block based on SP (in Auto and Cas modes).
OUT_ARRAY	None	Array of four entries (Y11, Y12, Y21, Y22) that defines the block output range starting and ending points used in the calculation of OUT_1 (Y11 and Y12) and OUT_2 (Y21 and Y22). Y11, Y12, Y21, and Y22 are standard industry terms that correlate to the DeltaV parameters OUT_ARRAY[1][1], OUT_ARRAY[2][1], OUT_ARRAY[3][1], and OUT_ARRAY[4][1], respectively.
SP	Supplied by input	The block's setpoint value.

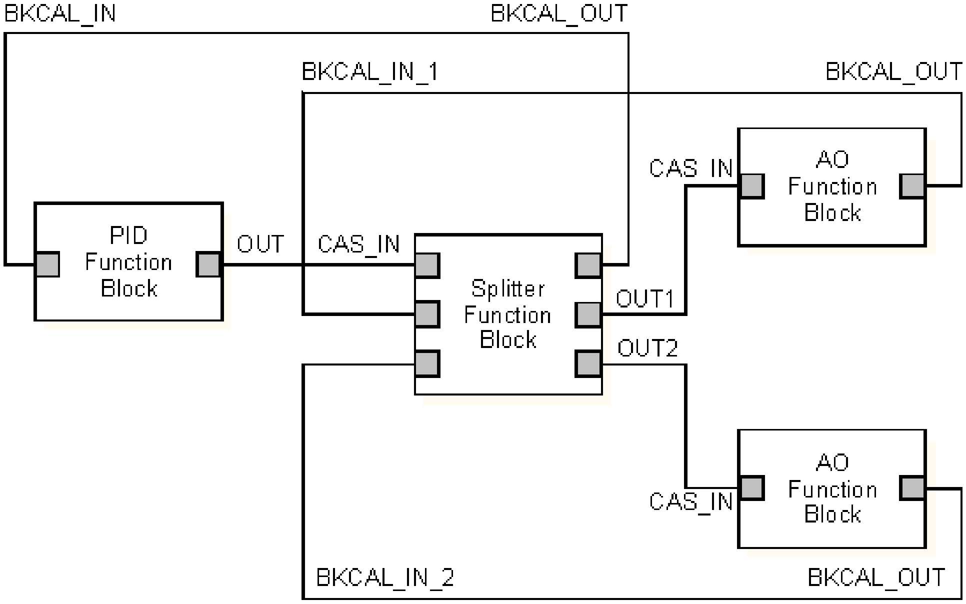
Parameter	Units	Description
SP_RATE_DN	Input units per second	Ramp rate at which downward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_RATE_UP	Input units per second	Ramp rate at which upward setpoint changes are acted on in Auto mode, in PV units per second. If the ramp rate is set to 0.0, then the setpoint is used immediately. For control blocks, rate limiting applies only in Auto. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_WRK	Input units	The working setpoint of the block. It is the result of setpoint limiting and setpoint rate of change limiting.
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*
STATUS_OPTS	None	Status options Allows you to select IFS if Bad CAS_IN.*
STRATEGY	None	The strategy filed can be used to help group blocks. This data is not checked or processed by the block.*

* These parameters are only visible when the function block is located in a Fieldbus device.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Splitter Function Block

You can use the Splitter function block to regulate multiple control loops or outputs. The following figure shows a function block diagram example when the output from a PID function block is split into two signals that are sent to Analog Output (AO) function blocks:



Splitter Function Block Diagram Example

The following examples illustrate typical uses for the Splitter function block.

Application Example: Split Range Control

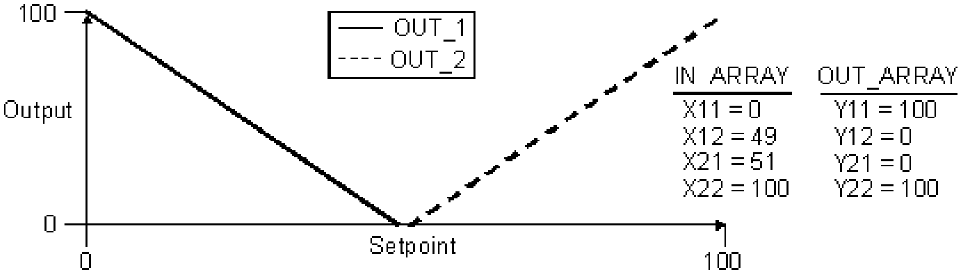
Assume two valves (one for heating and one for cooling) are driven by a single temperature controller. You use the Splitter function block between the controller and two Analog Output (AO) function blocks. When the controller requires heat, the AO block action for the cooling valve does not matter. When the AO block for the heating valve goes to Bad status, the controller is prevented from calling for more heat, but is allowed to call for cooling if required. For this example, you can set up the splitter characterization in the following manner:

Splitter Function Block Split Range Control Example Settings

Parameter	Setting
IN_ARRAY[1][1]	0
IN_ARRAY[2][1]	49
IN_ARRAY[3][1]	51
IN_ARRAY[4][1]	100
LOCKVAL	Hold
OUT_ARRAY[1][1]	100
OUT_ARRAY[2][1]	0

Parameter	Setting
OUT_ARRAY[3][1]	0
OUT_ARRAY[4][1]	100

This allows the controller to call for maximum cooling (negative heat) at 0 output and maximum heat at 100 output, with a gap of 2 around 50 to make sure that the heating and cooling valves are not both open at the same time. The following figure graphically represents the settings for this example:



Splitter Function Block Split Ranging Example

Application Example: Sequencing Control

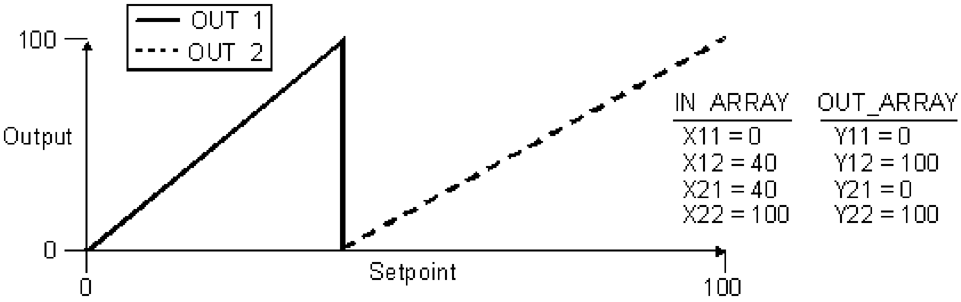
Assume you must use two valves for control because one valve does not provide enough range for a nonlinear control problem, such as pH control. The small valve controls an expensive reagent in the pH region near neutral; therefore, it must be closed when the pH is away from neutral. The big valve adds caustic, which is too strong to use near a neutral pH.

You can use the Splitter function block between the controller and two Analog Output (AO) blocks. As in the preceding example, one AO block can have Bad status without disturbing control when it is not the block required by the pH controller. For this example, you can set up the splitter characterization in the following manner:

Splitter Function Block Sequencing Control Example Settings

Parameter	Setting
IN_ARRAY[1][1]	0
IN_ARRAY[2][1]	40
IN_ARRAY[3][1]	40
IN_ARRAY[4][1]	100
LOCKVAL	OUT_ARRAY[1][1]
OUT_ARRAY[1][1]	0
OUT_ARRAY[2][1]	100
OUT_ARRAY[3][1]	0
OUT_ARRAY[4][1]	100

The following figure graphically represents the settings for this example:



Splitter Function Block Sequencing Control Example

Application Example: Cascade Fan-out

The cascade fan-out scheme is used when a cascade control strategy includes multiple secondary controllers. For example, assume a header pressure controller interacts with two secondary boiler controllers. You use the Splitter function block between the pressure controller and the two boiler master controllers. Header pressure control is maintained when one or both secondary Bias/Gain blocks are in Auto mode. When both secondary blocks are not in Auto mode, the pressure controller output is frozen and initializes to balance the first secondary block that is put into Auto mode.

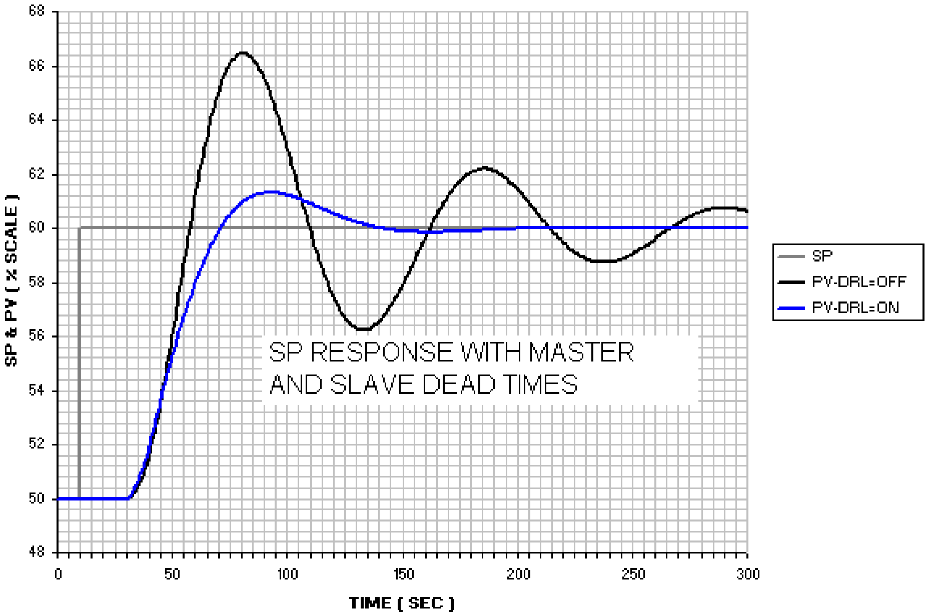
For this example, you can set up the splitter characterization in the following manner:

Splitter Function Block Cascade Fan-out Example Settings

Parameter	Setting
IN_ARRAY[1][1]	0
IN_ARRAY[2][1]	100
IN_ARRAY[3][1]	0
IN_ARRAY[4][1]	100
LOCKVAL	Hold
OUT_ARRAY[1][1]	0
OUT_ARRAY[2][1]	100
OUT_ARRAY[3][1]	0
OUT_ARRAY[4][1]	100
BAL_TIME	Four times the pressure controller integral time

Advanced Information - Splitter Function Block

Initialization is not automatic under all circumstances because some situations have conflicting needs. In general, when a control region is in trouble, the upstream controller is limited so it will not drive further into that region. When you want to move to the other active region, you can put the upstream controller into Man mode and move it, or you can drop the good downstream block out of Cas mode for one evaluation cycle and then restore Cas mode. This initializes the upstream controller to the remaining good region of control.

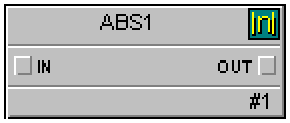


Math Blocks

This chapter contains information on mathematical function blocks in the DeltaV system.

Absolute Value (ABS) Function Block

The Absolute Value function block provides the absolute value of an integer or floating point input value. The block supports signal status propagation. There are no modes or alarm detection in the Absolute Value function block.



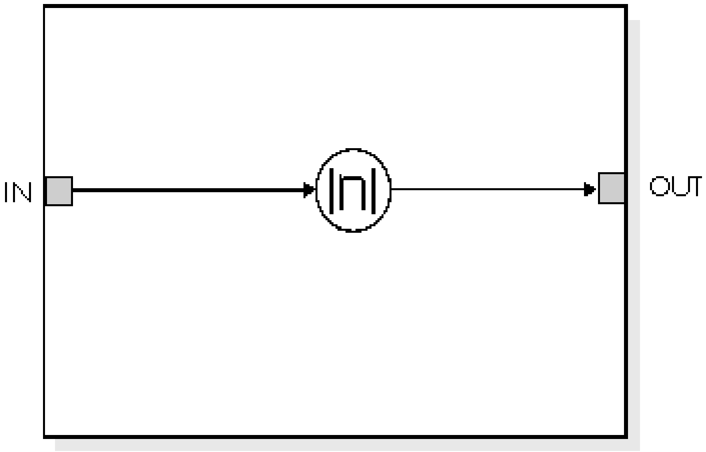
Absolute Value Function Block

IN is the input value and status to the block.

OUT is the output value and status.

Schematic Diagram - Absolute Value Function Block

The following figure shows the internal components of the Absolute Value function block:



Absolute Value Function Block Schematic Diagram

Block Execution - Absolute Value Function Block

The Absolute Value function block uses the following equation to calculate the output:

$$OUT = |IN|$$

Status Handling - Absolute Value Function Block

The output status is set to the status of the input.

Parameters - Absolute Value Function Block

The following table lists the system parameters for the Absolute Value function block:

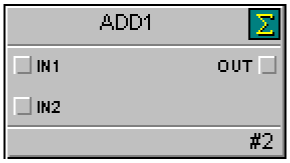
Absolute Value Function Block System Parameters

Parameter	Units	Description
IN	Determined by source	The analog input value and status.
OUT	Determined by IN	The analog output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Add Function Block

The Add function block sums the values of two to sixteen inputs and generates an output value. The block supports signal status propagation. There are no modes or alarm detection in the Add function block.



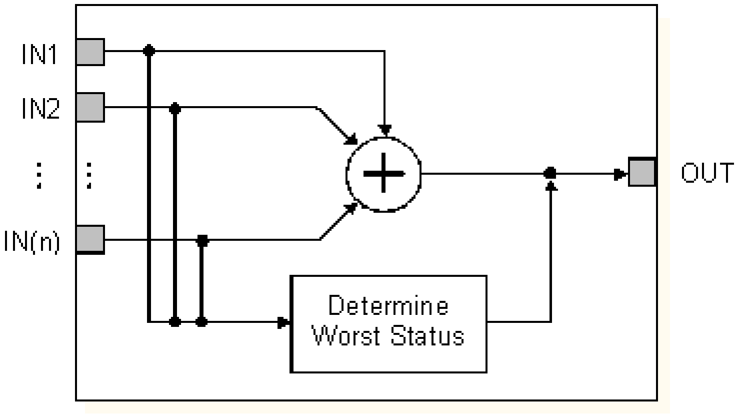
Add Function Block

IN1 through IN[n] are the input values and statuses to the block (as many as 16 inputs allowed).

OUT is the output value and status.

Schematic Diagram - Add Function Block

The following figure shows the internal components of the Add function block:



Add Function Block Schematic Diagram

Block Execution - Add Function Block

The number of inputs to the Add function block is an extensible parameter. The block default is two inputs. You add inputs by selecting the function block diagram, clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs. This creates additional input connectors for the block.

The Add function block uses the following equation to calculate the output:

$$OUT = \sum_{n=1}^{16} IN_n$$

The following table shows an example of Add function block outputs based on different input values:

Add Function Block Calculation Example

Parameter	Example 1	Example 2
IN1	1.0	1.5
IN2	3.7	2.5
IN3	—	0.2
IN4	—	3.1
OUT	4.7	7.3

Status Handling - Add Function Block

The output status is set to the worst status among the specified inputs.

Parameters - Add Function Block

The following table lists the system parameters for the Add function block:

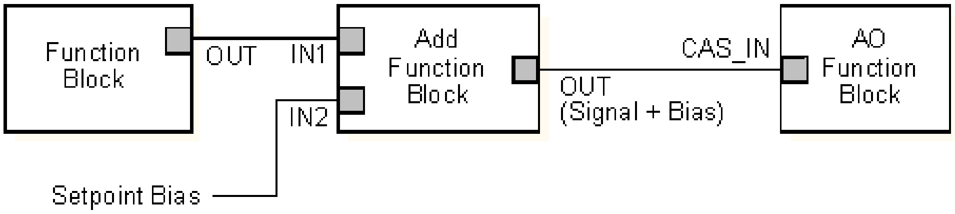
Add Function Block System Parameters

Parameter	Units	Description
IN1 to IN16	Determined by source	The analog input values and statuses. The number of inputs is an extensible parameter.
OUT	Determined by IN	The analog output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Add Function Block

The Add function block is used to sum multiple inputs. For example, you can use the Add block to compute the total flow of a manifold pipe stream. Or, you can use the block to calculate a setpoint bias setting based on logic performed in other function blocks, as in the following figure:



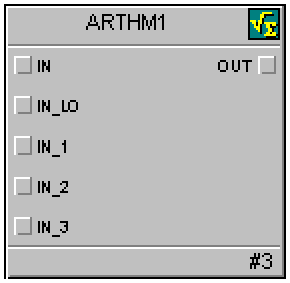
Add Function Block Application Example

Arithmetic Function Block

The Arithmetic function block provides the ability to configure a range extension function for a primary input and applies the nine different arithmetic types as compensation to or augmentation of the range extended input. All operations are selected by parameter and input connection.

The nine arithmetic functions are Flow Compensation Linear, Flow Compensation Square Root, Flow Compensation Approximate, Btu Flow, Traditional Multiply and Divide, Average, Summer, Fourth Order Polynomial, and Simple HTG Compensate Level.

This Arithmetic function block supports mode control (Auto, Man, OOS). There is no standard alarm detection in this block.

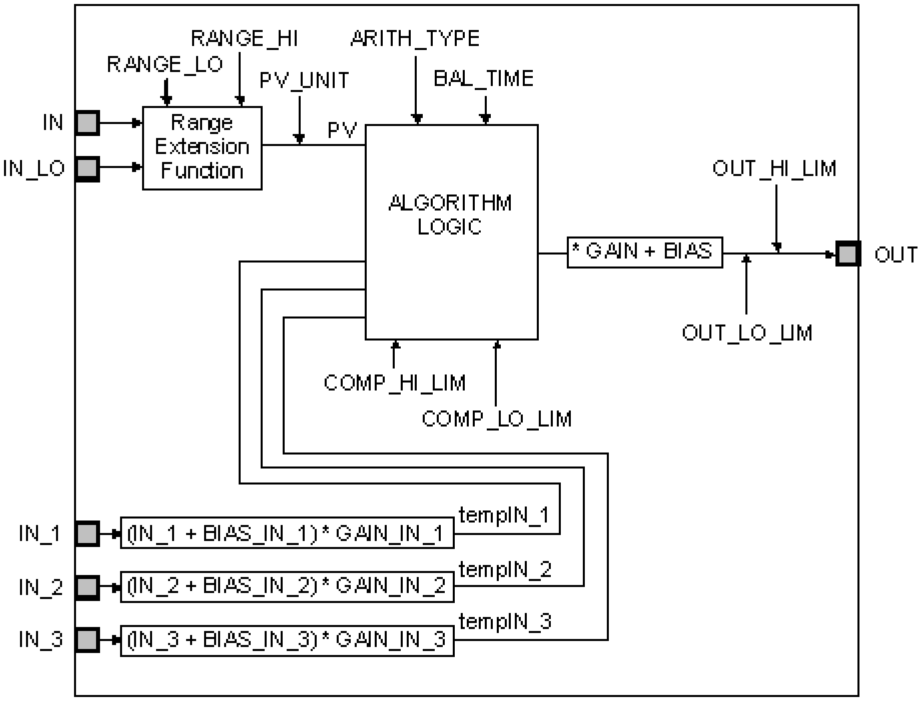


Arithmetic Function Block

Note The value and usage of IN_1 through IN_3 depend on the math function selected. For example, Flow Compensation Approximate uses IN_1, IN_2, and IN_3, whereas Flow Compensation Linear uses **only** IN_1 and IN_2. Refer to the Advanced Topics - Arithmetic Types topic for a complete list of all math functions and formulas used.

Schematic Diagram - Arithmetic Function Block

The following diagram shows the internal components of the Arithmetic function block:



Arithmetic Function Block Schematic Diagram

Block Execution - Arithmetic Function Block

The Arithmetic function block provides range extension and compensation through nine arithmetic types.

There are two inputs (IN and IN_LO) used in calculating PV. Inputs used to form the PV must come from devices with the desired engineering units. PV is then combined with up to three inputs (IN_1, IN_2, and IN_3) through the user-selected compensation function (ARITH_TYPE) to calculate the value of func. A gain is applied to func, and then a bias is added to get the value PRE_OUT. In AUTO, PRE_OUT is used for OUT.

Range Extension and Calculation of PV

When both IN and IN_LO are usable, the following formula is applied to calculate range extension for PV:

PV = G*IN+(1-G)*IN_LO

G is zero for IN less than RANGE_LO. It is one when IN is greater than RANGE_HI. It is interpolated from zero to one over the range of RANGE_LO to RANGE_HI. If the status of IN_LO is unusable and IN is usable and greater than RANGE_LO, then G will be set to one.

Compensation Input Calculations

For each of the inputs (IN_1, IN_2, IN_3), there is a gain and bias. The bias can be used to correct for absolute temperature or pressure. The gain can be used to normalize terms within a square root function. The compensation terms (t) are calculated as follows:

- When IN_(k) is usable:
  t(k) = GAIN_IN(k)*(BIAS_IN(k) = IN_ (k))
- When IN_(k) is not usable, then t(k) gets the value of the last t(k) computed with a usable input.

Application Example

The Arithmetic block can be used for pressure and temperature compensation of a gas stream. IN is the measured flow from OUT of an Analog Input function block. IN_1 is the measured pressure, and IN_2 is the measured temperature, both from OUT of an Analog Input function block. If the flowmeter is a differential pressure type, the flow must be linearized in the instrument or in the AI function block.

ARITH_TYPE is **flow compensation linear** if a mass flowmeter is used or **flow compensation square root** if the flowmeter is differential pressure.

BIAS_IN_1 is a factor to convert measured pressure to absolute pressure. If pressure units are PSIG, for example, enter 14.696 for BIAS_IN_1.

GAIN_IN_1 is 1 / (reference pressure + factor to convert to absolute pressure). Reference pressure is the calibration pressure for the flowmeter. If the calibration pressure is 50 PSIG, for example, enter 0.01546 for GAIN_IN_1.

BIAS_IN_2 is a factor to convert measured temperature to absolute temperature. If temperature units are degF, for example, enter 459.7 for BIAS_IN_2.

GAIN_IN_2 is 1 / (reference temperature + factor to convert to absolute temperature). Reference temperature is the calibration temperature for the flowmeter. If the calibration temperature is 350 degF, for example, enter 0.001235 for GAIN_IN_2.

Modes - Arithmetic Function Block

The Arithmetic function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The target mode of a block might be restricted to one or more of the supported modes.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Arithmetic Function Block

IN_x Use Bad

IN_x Use Uncertain

IN_LO Use Uncertain

IN Use Uncertain

If the status of IN is unusable and IN_LO is usable and less than RANGE_HI, then G will be set to zero. (For a definition of G, refer to the Block Execution - Arithmetic Function Block topic.) In each case, the PV will have a status of Good until the condition no longer applies. Otherwise, the status of IN_LO is used for the PV if G is less than 0.5, while IN is used for G greater than or equal to 0.5.

The status of PV is copied to PRE_OUT.

For complete descriptions of supported input options, refer to the Input Options topic.

Parameters - Arithmetic Function Block

The following table lists the parameters for the Arithmetic function block.

Arithmetic Function Block Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ARITH_TYPE	None	The set of nine arithmetic functions applied as compensation to or augmentation of the range extended input.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for a block value to match an input, output, or calculated value or the time for dissipation of the internal balancing bias.
BIAS	None	The bias value
BIAS_IN_1	None	The bias value for IN_1

Parameter	Units	Description
BIAS_IN_2	None	The bias value for IN_2
BIAS_IN_3	None	The bias value for IN_3
BLOCK_ERR	None	The summary of active error conditions associated with the block. The possible block errors are Block configuration error, Simulate active, Local override, Input failure/process variable has Bad status, Output failure, Readback failed, Out of Service, and Other. Each function block reports none or a subset of these error conditions.
COMP_HI_LIM	EU of PV_SCALE	Determines the high limit of the compensation input
COMP_LO_LIM	EU of PV_SCALE	Determines the low limit of the compensation input
GAIN	None	The proportional gain (multiplier) value
GAIN_IN_1	None	The proportional gain (multiplier) value for IN_1
GAIN_IN_2	None	The proportional gain (multiplier) value for IN_2
GAIN_IN_3	None	The proportional gain (multiplier) value for IN_3
IN	Determined by source or EU of PV_SCALE or EU of IN_SCALE	The analog input value and status. The number of inputs is an extensible parameter in some function blocks.
IN_1	Determined by supplying block or source	The first analog input value and status
IN_2	Determined by supplying block or source	The second analog input value and status
IN_3	Determined by supplying block or source	The third analog input value and status
IN_LO	None	The value used for the input whenever IN is below range

Parameter	Units	Description
INPUT_OPTS	None	Allow you to set the options for using IN and IN_LO when Uncertain. They also allow you to set the options for using IN_1, IN_2 and IN_3 when any are either Bad or Uncertain. When a particular option is **not** selected and the status is Bad or Uncertain, the last usable value is used. A value is usable if the status is Good or if the status is not Good but the input option for that status (Bad or Uncertain) is selected to use it. To change the input options while using the on-line view, first change the mode to OOS.
MODE	None	The mode record of the block. MODE contains the actual, target, permitted, and normal modes. In some function blocks, this parameter is used to request and show the source of the setpoint, the source of the output, and/or the block operating state.
OUT	EU of OUT_SCALE or Percent or EU of IN	The analog output value and status. The number of outputs is an extensible parameter in some blocks.
OUT_HI_LIM	EU of OUT_SCALE or Supplied by IN	The maximum output value allowed
OUT_LO_LIM	EU of OUT_SCALE or Supplied by IN	The minimum output value allowed
PRE_OUT	EU of PV_SCALE	The calculated result of the selected math function after limiting
PV	EU of OUT or EU of PV_SCALE or EU of IN_SCALE	The process variable used in block execution and alarm limit detection
RANGE_HI	EU of PV_SCALE	The high limit for IN
RANGE_LO	EU of PV_SCALE	The low limit for IN. If IN is less than RANGE_LO, then IN_LO is used.

Advanced Topics - Arithmetic Types

The ARITH_TYPE parameter determines how PV and the compensation terms (t) are combined. The user can select from nine commonly used math functions, which are depicted below. COMP_HI and COMP_LO are compensation limits.

- Flow Compensation Linear

$$func = PV * f$$

$$\overset{COMP\_HI}{\underset{COMP\_LO}{f = \frac{t(1)}{t(2)}}}$$

- Flow Compensation Square Root

$$func = PV * f$$

$$\overset{COMP\_HI}{\underset{COMP\_LO}{f = \sqrt{\frac{t(1)}{t(2) * t(3)}}}}$$

If there is a divide by zero and the numerator is positive, f is set to COMP_HI; if the numerator is negative, then f is set to COMP_LO. The square root of a negative value will equal the negative of the square root of the absolute value. Imaginary roots are not supported.

- Flow Compensation Approximate

$$func = PV * f$$

$$\overset{COMP\_HI}{\underset{COMP\_LO}{f = \sqrt{t(1) * t(2) * t(3)^2}}}$$

- Btu Flow

$$func = PV * f$$

$$\overset{COMP\_HI}{\underset{COMP\_+LO}{f = t(1) - t(2)}}$$

- Traditional Multiply and Divide

$$func = f * PV$$

$$\overset{COMP\_HI}{\underset{COMP\_LO}{f = \frac{t(1)}{t(2)} + t(3)}}$$

If there is a divide by zero and numerator is positive, f will be limited to COMP_HI; if the numerator is negative, f will be limited to COMP_LO.

- Average

$$\{\mathrm{Sum} = \mathrm{PV.Val}\ ;\ n = 1\}$$

$$\mathrm{For}\ k = 1{,}3\ \{\mathrm{sum} = \mathrm{sum} + t(k);\ n = n + 1\}\ \mathrm{EndFor}$$

$$\mathrm{func} = \frac{\mathrm{sum}}{n}$$

Compensation inputs that are not usable are not included in the calculation. PV is always included.

- Summer

$$\mathrm{sum} = \mathrm{PV}$$

$$\mathrm{For}\ k = 1{,}3\ \{\mathrm{sum} = \mathrm{sum} + t(k)\}\mathrm{EndFor}$$

$$\mathrm{func} = \mathrm{sum}$$

Compensation inputs that are not configured are not used in the calculation. PV is always used.

- Fourth Order Polynomial

$$\mathrm{func} = \mathrm{PV} + t(1)^2 + t(2)^3 + t(3)^4$$

- Simple HTG Compensate Level

$$\mathrm{func} = \frac{\mathrm{PV} - t(1)}{\mathrm{PV} - t(2)}$$

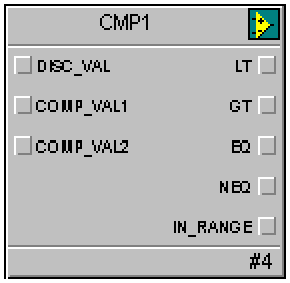
If there is a divide by zero and the numerator is positive, func will be limited to COMP_HI; if the numerator is negative, func will be limited to COMP_LO.

Comparator Function Block

The Comparator (CMP) function block allows you to compare two values (DISC_VAL and COMP_VAL1) and set a Boolean output based on that comparison for the LT (Less Than), GT (Greater Than), EQ (Equal To), NEQ (Not Equal) outputs.

Additionally, the Comparator function block compares the DISC_VAL against the range defined by COMP_VAL2 and COMP_VAL1 to determine the Boolean output, IN_RANGE.

The Comparator function block does not have any modes or alarm detection.

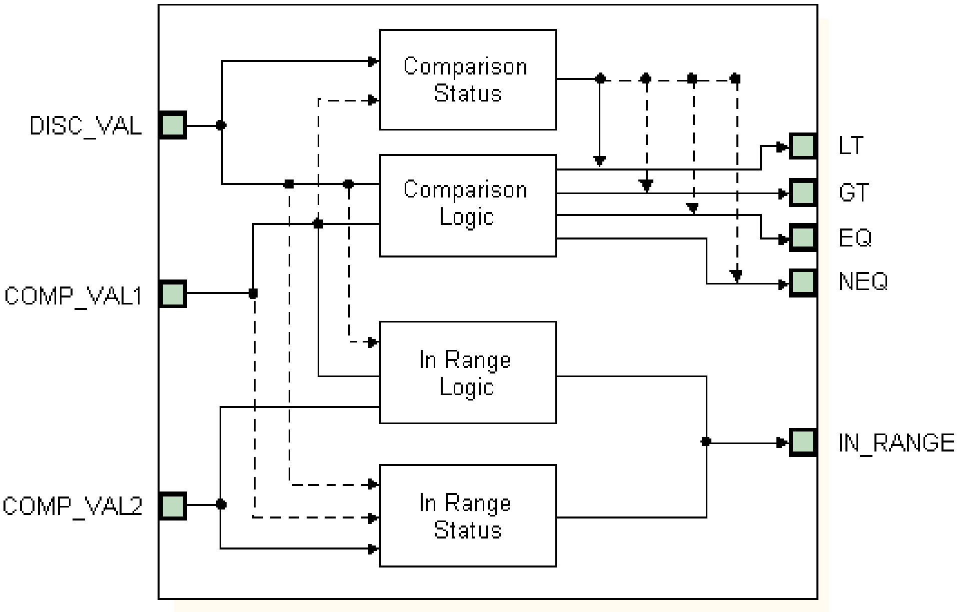


Comparator Function Block

Note If you are using a Condition function block with one check, use this block instead.

Schematic Diagram - Comparator Function Block

The following figure shows the internal design of the Comparator function block.



Comparator Function Block Schematic Diagram

Block Execution - Comparator Function Block

The Comparator function block has two block calculations: the comparison calculation and the status propogation. Both are described here.

Comparison Calculation

The Comparator function block takes the DISC_VAL input and performs a compare operation with COMP_VAL1, the primary comparison value. Based on the relationship between DISC_VAL and COMP_VAL1, the LT, GT, EQ, NEQ outputs will be set to 0 (False) or 1 (True). A secondary comparison is done to verify if the DISC_VAL is within the range of COMP_VAL1 to COMP_VAL2. If DISC_VAL is within this range, then the IN_RANGE output will be set to 1 (True), otherwise 0 (False).

Status Propagation

Bad status on any of the input values is propagated to the output. If the DISC_VAL has a bad status, all outputs will reflect this bad status. If DISC_VAL has good status but COMP_VAL1 or COMP_VAL2 has bad status, then the outputs associated with the bad input are also set to bad. The status calculation is totally independent of the comparison calculations.

The following table shows an example of the Comparator function block outputs based on different input values.

Sample Comparator Function Block Outputs

Parameter	Example 1	Example 2	Example 3
DISC_VAL	2.25	-233.0	37.5
COMP_VAL1	15.0	-200.0	37.5
COMP_VAL2	1.0	0.0	10.0
LT	1	1	0
GT	0	0	0
EQ	0	0	1
NEQ	1	1	0
IN_RANGE	1	0	1

Status Handling - Comparator Function Block

The output status is set to the worst status among the inputs.

For example, if the status on COMP_VAL1 is bad, the statuses on LT, GT, EQ, and NEQ are all set to BAD. Also, if the status on COMP_VAL1 or COMP_VAL2 is BAD, then the status on IN_RANGE is BAD.

Parameters - Comparator Function Block

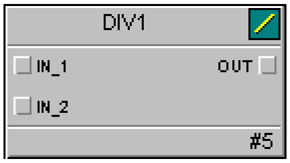
The following table lists the system parameters for the Comparator function block.

Comparator Function Block Parameters

Parameter	Units	Description
COMP_VAL1	Determined by source	The first comparative value. Used to compare with DISC_VAL in calculating EQ, GT, LT, and NEQ. Used as part of the range that DISC_VAL compares against to calculate IN_RANGE.
COMP_VAL2	Determined by source	The second comparative value. Used as part of the range that DISC_VAL compares against to calculate IN_RANGE.
DISC_VAL	Determined by source	The input comparison value. Used with COMP_VAL1 in calculating EQ, GT, LT, and NEQ. Compared against the range of COMP_VAL1 - COMP_VAL2 to calculate IN_RANGE.
EQ	None	Equal to. Compares COMP_VAL1 with DISC_VAL. Discrete output (True=1/False=0).
GT	None	Greater Than. Compares COMP_VAL1 with DISC_VAL. Discrete output (True=1/False=0).
IN_RANGE	None	In range. Compares DISC_VAL against the range of COMP_VAL2 and COMP_VAL1.
LT	None	Less Than. Compares COMP_VAL1 with DISC_VAL. Discrete output (True=1/False=0).
NEQ	None	Not Equal To. Compares COMP_VAL1 with DISC_VAL. Discrete output (True=1/False=0).

Divide Function Block

The Divide (DIV) function block divides one input value by another input value and generates an output value. The block supports signal status propagation. There are no modes or alarm detection in the Divide function block.



Divide Function Block

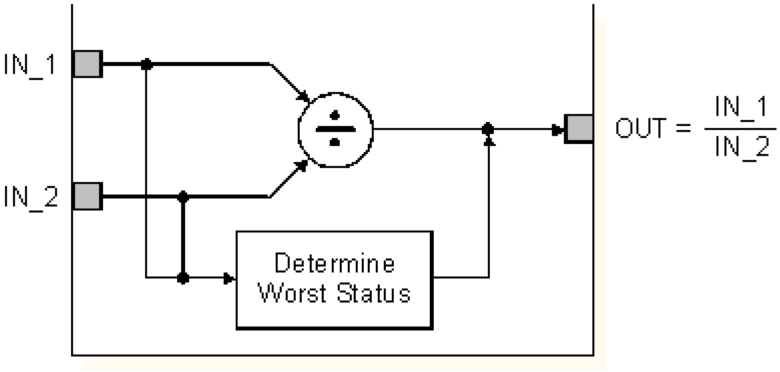
IN_1 is the input value and status to be divided (numerator).

IN_2 is the input value and status of the divisor.

OUT is the output value and status.

Schematic Diagram - Divide Function Block

The following figure shows the internal components of the Divide function block:



Divide Function Block Schematic Diagram

Block Execution - Divide Function Block

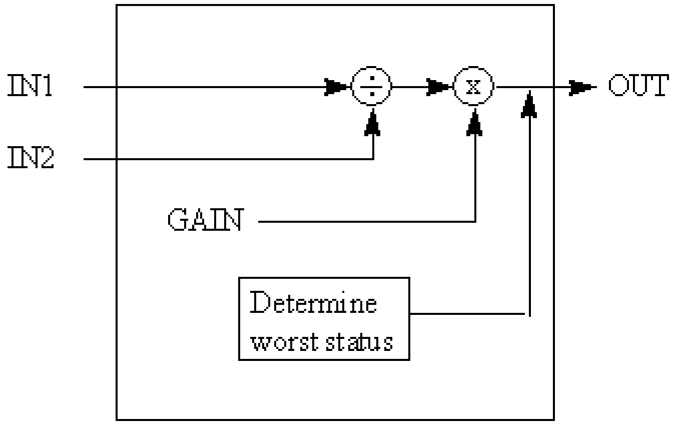
The Divide function block uses the following equation to calculate the output:

$$OUT = \left(\frac{IN\_1}{IN\_2}\right)$$

When IN_2 equals zero, OUT is set to the maximum floating point value.

Status Handling - Divide Function Block

The output status is set to the worst status among the inputs.



Divide Function Block Worst Status Diagram

Parameters - Divide Function Block

The following table lists the system parameters for the Divide function block.

Divide Function Block System Parameters

Parameter	Units	Description
IN_1	Determined by source	The analog input value and status to be divided (numerator)
IN_2	Determined by source	The analog input value and status of the divisor
OUT	EU of IN	The analog output value and status

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Divide Function Block

The Divide function block is useful for conversion calculations.

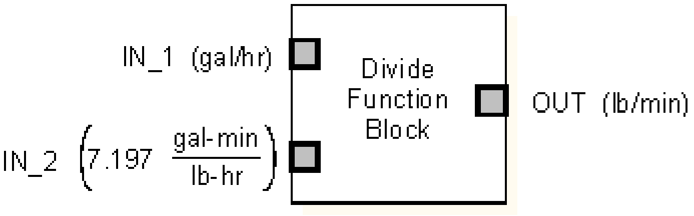
Application Example: Unit Conversion

Assume a transmitter measuring water flow is calibrated in gallons per hour, but the operator needs the information in pounds per minute. You can use the Divide function block to calculate the correct units, as in the following equations:

$$OUT = \frac{lb}{min} = \frac{(gal/hr)}{\left(\frac{gal}{8.337\ lb} \times \frac{60min}{hr}\right)} = \frac{IN\_1}{IN\_2}$$

$$IN\_2 = 7.197\ \frac{gal - \min}{lb - hr}$$

The following figure is the function block diagram for this example:



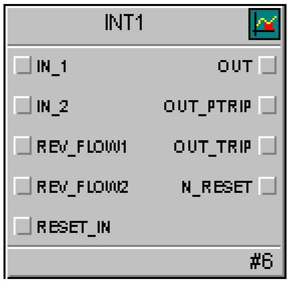
Divide Function Block Diagram Example

Integrator Function Block

The Integrator (INT) function block integrates one variable or the sum or difference between two variables over time. The block compares the integrated or accumulated value to pre-trip and trip limits and generates discrete output signals when the limits are reached. This function block can also be used as a totalizer.

You choose one of six integrator types that determine whether the integrated value increases from 0 or decreases from the trip value. The block has two inputs and can integrate positive, negative, or net flow. This capability is useful to calculate volume or mass variation in vessels or as an optimization tool for flow ratio control.

The Integrator function block supports mode control, demand reset, a reset counter, and signal status calculation. There are no standard alarms in this function block. Custom alarms are supported.



Integrator Function Block

IN_1 is the first input value and status. Specify the rate time base for IN_1 with TIME_UNIT1.

IN_2 is the second input value and status. Specify the rate time base for IN_2 with TIME_UNIT2.

REV_FLOW1 is the discrete input that specifies whether IN_1 is positive or negative.

REV_FLOW2 is the discrete input that specifies whether IN_2 is positive or negative.

RESET_IN is the discrete input that resets the integrator and holds reset until released.

OUT is the integration output value and status. OUT equals TOTAL when the integration type counts up (0 to SP). OUT equals TOTAL when the integration type counts down (SP to 0).

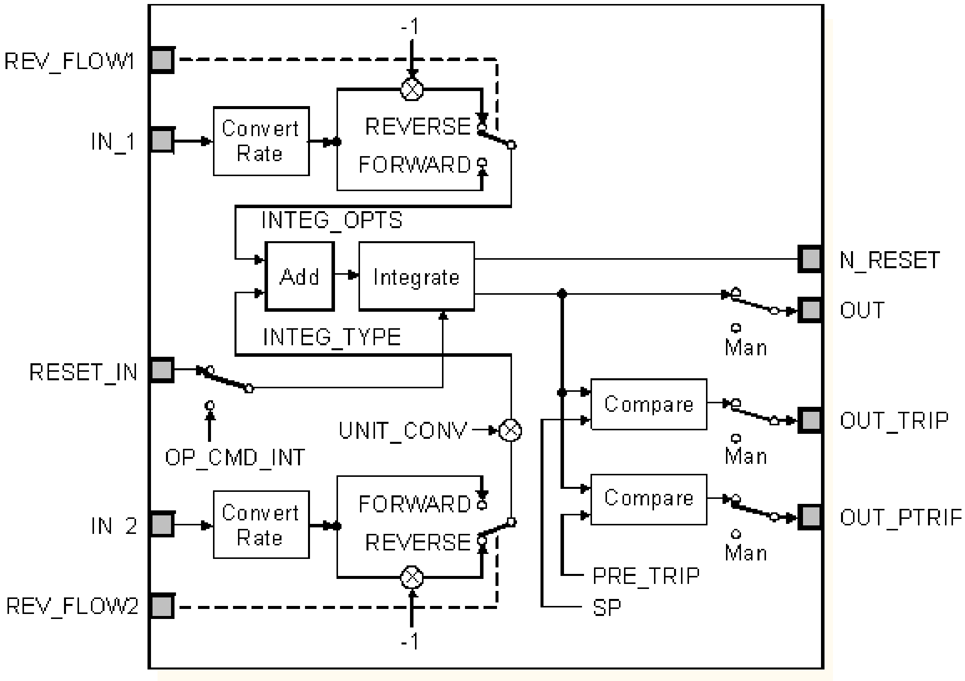
OUT_PTRIP is a discrete value that is set when the pre-trip limit value is reached. OUT_PTRIP remains latched until OUT_TRIP latches, then OUT_PTRIP unlatches.

OUT_TRIP is a discrete value that is set when the trip target value is reached. OUT_TRIP latches for only one scan, then unlatches.

N_RESET is the number of times the integrator is initialized or reset.

Schematic Diagram - Integrator Function Block

The following figure shows the internal components of the Integrator function block:



Integrator Function Block Schematic Diagram

Block Execution - Integrator Function Block

The Integrator function block integrates a variable over time. The integrated or accumulated value (OUT) is compared to pre-trip and trip limits. When the limits are reached, discrete output signals are generated (OUT_PTRIP and OUT_TRIP). You choose whether the integrated value increases from 0 or decreases from the trip value (SP). The block has two inputs (IN_1 and IN_2) and can integrate positive, negative, or net flow.

The transfer equation used in the Integrator function block is:

$$\mathit{Current\ Integral} = (\Delta t) \times (x + y) + \mathit{OUT}[t - 1]$$

where

t = the elapsed time since previous cycle, in seconds

x = the converted IN_1 value (based on the options you configure)

y = the converted IN_2 value (based on the options you configure) or 0 when you select not to use a second input

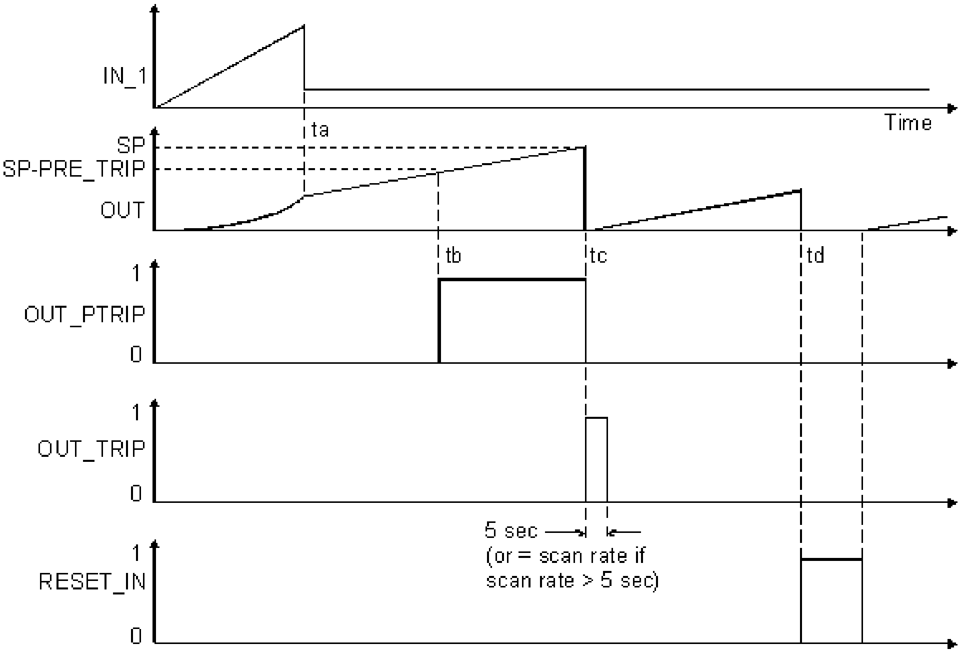
OUT[t-1] = the value of OUT from the previous cycle.

You can choose integration type options that define the integrate up, integrate down, and reset characteristics of the block. Refer to the Integration Types section below for information on these options. When you select the SP to 0 - auto reset or SP to 0 - demand reset integration type option, the following applies:

OUT = SP - Integral

For all other integration types:

OUT = Integral



Integrator Response to Changing Input

Note In the above graph, read PRE_TRIP as the distance below SP where OUT_PRETRIP is set.

You can specify how the block executes by configuring input flow and rate time variables, integration type and carryover options as well as trip and pre-trip action.

In the above example, IN_1 and RESET are inputs. IN_2 is not connected. Integration proceeds from 0 to SP. From *t*=0 until *ta*, there is ramp input. For *t*>*ta*, the input drops to a lower constant value.

Note the output produced by integrating this input pattern. The ramp input integrates into a parabolic curve between *t*=0 and *t*=*ta*. Beyond *ta*, the constant input integrates into a ramp output with a constant slope. At *tb*, OUT reaches the level SP-PRE_TRIP. OUT_PTRIP transitions from 0 to 1. It remains at 1 until *tc*, where OUT reaches SP. At this point, OUT_PTRIP returns to 0 while OUT_TRIP transitions from 0 to 1. OUT_TRIP remains at 1 for 5 seconds or for the duration of the scan rate if it is greater than 5 seconds. It then returns to 0.

At *tc*, when OUT reaches SP, OUT is reset to 0, and the integration resumes anew. At *td*, the RESET_IN input transitions to 1, interrupts the integration, and resets OUT to 0.

Specifying Rate Time Base

The time unit parameters (TIME_UNIT1 and TIME_UNIT2) specify the rate time base of the inputs (IN_1 and IN_2, respectively). The block uses the following equations to compute the integration increment:

$$x = \frac{\text{IN\_1}}{\text{TIME\_UNIT1}}$$

$$y = \frac{\text{IN\_2}}{\text{TIME\_UNIT2}}$$

where

x = the converted IN_1 value (based on the options you configure)

y = the converted IN_2 value (based on the options you configure) or 0 when you select not to use a second input.

The block supports the following options for TIME_UNIT1 and TIME_UNIT2:

Seconds (= 1 second)

Minutes (1 minute = 60 seconds)

Hours (1 hour = 3600 seconds)

Days (1 day = 86400 seconds)

Setting Reverse Flow at the Inputs

Reverse flow is determined by either of the following:

- The sign of the value at IN_1 or IN_2
- The discrete inputs REV_FLOW1 and REV_FLOW_2

When the REV_FLOW input is True, the block interprets the associated IN value as negative.

Calculating Net Flow

Net flow is calculated by adding the increments calculated for each IN. When ENABLE_IN_2 is False, the increment value for IN_2 is considered 0. When ENABLE_IN_2 is True, the value of IN_2 is used in the calculation.

You can determine the net flow direction that is to be included in the integration by configuring the Flow Forward and Flow Reverse integration options parameter (INTEG_OPTS). When Flow Forward is True, positive increments are included. When Flow Reverse is True, negative increments are included. When both Flow Forward and Flow Reverse are True, both positive and negative increments are included.

Integration Types

The integration type parameter (INTEG_TYPE) defines the integrate up, integrate down, and reset characteristics of the block. You can choose from the following options:

0 to SP - auto reset at SP – Integrates from 0 to the setpoint (SP) and automatically resets when SP is reached.

0 to SP - demand reset – Integrates from 0 to SP. When OUT is within PRE_TRIP of SP, P_TRIP is set. When OUT reaches SP, TRIP is set and the block continues integrating beyond SP.

SP to 0 - auto reset at SP – Integrates from SP to 0 and automatically resets when 0 is reached.

SP to 0 - demand reset – Integrates from SP to 0. When OUT is within PRE_TRIP of 0, then P_TRIP is set. When OUT reaches SP, TRIP is set and the block continues integrating below 0.

0 to ? - periodic reset – Counts upward and resets periodically. The period is set by the CLOCK_PER parameter.

0 to ? - demand reset – Counts upward and is reset when RESET_IN or OP_CMD_INT transitions to True.

0 to ? - periodic & demand reset – Counts upward and is reset periodically or when RESET_IN or the OP_CMD_INT transitions to True.

The following table summarizes the trip, pre-trip and reset conditions for each of the mutually exclusive INTEG_TYPE options:

Summary of INTEG_TYPE Conditions

INTEG_TYPE	OUT	Automatic Reset Condition	Pre-Trip Condition	RESET_IN Reset	OP_CMD_INT Reset
0 to SP - auto reset at SP	TOTAL	$\begin{pmatrix}\text{SP} \geq 0\ \text{And}\\ \text{TOTAL} \geq \text{SP}\end{pmatrix}$	$\left(\begin{matrix}\text{SP} \geq 0\ \text{AND}\\ \begin{pmatrix}\text{PRE\_TRIP} \geq\\ \text{SP-TOTAL}\end{pmatrix}\end{matrix}\right)$	Yes	Yes
0 to SP - demand reset	TOTAL	None	$\left(\begin{matrix}\text{SP} \geq 0\ \text{And}\\ \begin{pmatrix}\text{PRE\_TRIP} \geq\\ \text{SP-TOTAL}\end{pmatrix}\end{matrix}\right)$	Yes	Yes
SP to 0 -auto reset at SP	SP-TOTAL	$\begin{pmatrix}\text{SP} \geq 0\ \text{And}\\ \text{TOTAL} \geq \text{SP}\end{pmatrix}$	$\left(\begin{matrix}\text{SP} \geq 0\ \text{And}\\ \begin{pmatrix}\text{PRE\_TRIP} \geq\\ \text{SP-TOTAL}\end{pmatrix}\end{matrix}\right)$	Yes	Yes
SP to 0 - demand reset	SP-TOTAL	None	$\left(\begin{matrix}\text{SP} \geq 0\ \text{And}\\ \begin{pmatrix}\text{PRE\_TRIP} \geq\\ \text{SP-TOTAL}\end{pmatrix}\end{matrix}\right)$	Yes	Yes
0 to ? - periodic reset	TOTAL	$\begin{pmatrix}\text{TimeSince}\\ \text{Last Reset} >\\ \text{CLOCK\_PER}\end{pmatrix}$	None	No	Yes
0 to ? - demand reset	TOTAL	None	None	Yes	Yes
0 to ? - periodic & demand reset	TOTAL	$\begin{pmatrix}\text{TimeSince}\\ \text{Last Reset} >\\ \text{CLOCK\_PER}\end{pmatrix}$	None	Yes	Yes

Trip and Pre-trip Action

When the integration value reaches SP – PRE_TRIP (or 0 + PRE_TRIP if the integrator runs from SP to 0), OUT_PTRIP is set. When the integration value reaches the trip target value (SP or 0), OUT_TRIP is set. OUT_PTRIP remains set until integration is reset.

Integration Carryover

You can enable the Carry integration option (INTEG_OPTS) to carry the excess past the trip point into the next integration cycle as the initial value of the integrator when either of the following integration types is set:

- 0 to SP - auto reset at SP
- SP to 0 - auto reset at SP

Note that there must be at least five seconds between resets. SP must be large enough so that it takes more than five seconds for OUT to reach SP.

Modes - Integrator Function Block

The Integrator function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

The integrator initializes with the value in OUT when the mode changes from Man to Auto. The Man, Auto, and OOS modes can be configured as permitted modes for operator entry.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Integrator Function Block

Block execution is impacted by the status of IN_1 and IN_2. The increment is added to the accumulated value when the status of IN_1 (and IN_2 when applicable) is Good. If the status is not Good, the last Good status value is used to calculate the increment to be added to the accumulated value.

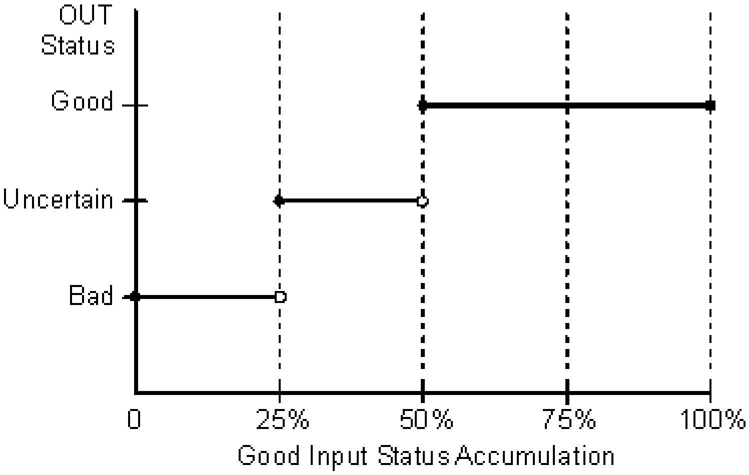
The output status calculation is based on the accumulation of input statuses. The calculation includes the accumulation for only the worst status of the two input channels when IN_2 is enabled.

Output status is determined by the integration of the absolute value of bad input values relative to integration of absolute value of total input values. An input status of uncertain is treated as bad for the quality calculation.

The output status is determined with the following logic:

- When the ratio of bad to total integrated absolute values is greater than 75%, OUT status is set to bad.
- When the ratio of bad to total integrated absolute values is between 50% and 25% OUT status is set to Uncertain.
- When the ratio of bad to total integrated absolute values is less than 25% then OUT status is set to good.

The following figure illustrates output status designations:



Integrator Function Block Output Status Determination

Parameters - Integrator Function Block

The following table lists the system parameters for the Integrator function block:

Integrator Function Block System Parameters

Parameter	Units	Definition
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ABSTOTAL	EU of OUT	The sum of the integrated absolute values of input.
ALERT_KEY	None	The identification number of the plant unit. This information can be used in the host for sorting alarms, and so on.*
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).

Parameter	Units	Definition
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block error for the Integrator function block is Out of Service.
CLOCK_PER	Seconds	Specifies the period for the time-based integration and reset cycle.
ENABLE_IN_2	None	Specifies whether or not IN_2 is used in the integration (True [1] = use IN_2, False [0] = do not use IN_2).
GOOD_LIM	Percent	Sets the good limit for PCT_INCL. If PCT_INCL is above this limit, then OUT receives the status, GOOD.
IN_1	Set by TIME_UNIT1	The first input value and status.
IN_2	Set by TIME_UNIT2	The second input value and status.
INTEG_OPTS	None	Integration options. Specify whether positive and/or negative net values are integrated and specifies whether carryover to the next cycle is used. The integration options are Flow Forward, Flow Reverse, and Carry.
INTEG_TYPE	None	Integration type. Allows you to specify integration and reset behavior. The integration types are 0 to SP - auto reset at SP, 0 to SP - demand reset, SP to 0 - auto reset at SP, SP to 0 - demand reset, 0 to ? - periodic reset, 0 to ? - demand reset, and 0 to ? - periodic and demand reset.
MODE	None	Parameter used to show and set the block operating state.
N_RESET	None	The number of times the integrator has been reset.
OP_CMD_INT	None	The operator command to reset the integrator. Use either 0 (false/no reset) or 1 (true/reset).
OUT	EU of IN	The integration output value and status. OUT equals TOTAL when the integration type counts up (0 to SP). OUT equals TOTAL when the integration type counts down (SP to 0).
OUTAGE_LIM	None	The maximum tolerated duration for power failure.*
OUT_PTRIP	None	Discrete output that is set True when the integrated value comes within PRE_TRIP of SP or 0.
OUT_TRIP	None	Discrete output that is set when the integrated value reaches the trip target (SP or 0).

Parameter	Units	Definition
PCT_INCL	Percent	Indicates the percentage of integrated values with a GOOD status.
PRE_TRIP	EU of OUT	The distance before the trip limit where OUT_PTRIP is set. Note that if PRE_TRIP =0, OUT_PTRIP is never set.
PULSE_VAL1	None	The factor used to convert IN_1 counts to engineering unit per pulse.
PULSE_VAL2	None	The factor used to convert IN_2 counts to engineering unit per pulse.
RESET_IN	None	Discrete input that resets the integrator.
REV_FLOW1	None	Discrete input that designates IN_1 as positive or negative.
REV_FLOW2	None	Discrete input that designates IN_2 as positive or negative.
RTOTAL	Application defined	The accumulated absolute value of all inputs with rejected statuses (BAD or UNCERTAIN).
SABSTOTAL	Application defined	The value of ABSTOTAL immediately before ABSTOTAL is reset.
SP	EU of OUT	The block's setpoint value.
SRTOTAL	Application defined	The value of RTOTAL immediately before RTOTAL is reset.
SSP	None	The value of SP immediately before SP is reset.
STATUS_OPTS	None	Selectable options for status handling and processing.
STOTAL	Application defined	The value of TOTAL immediately before TOTAL is reset.
STRATEGY	None	The strategy filed can be used to help group blocks. This data is not checked or processed by the block.*
ST_REV	None	The revision level of the static data associated with the function block. To support tracking changes in static parameter fields, the associated block's static revision parameter is incremented each time a static parameter field value is changed. Also, the associated block's static revision parameter is incremented if a static parameter field is written but the value is not changed.*
TIME_UNIT1	Selectable time unit	Specifies IN_1 rate time base (seconds, minutes, hours, days).
TIME_UNIT2	Selectable time unit	Specifies IN_2 rate time base (seconds, minutes, hours, days).
TOTAL_SP	None	The setpoint for a batch totalization.*

Parameter	Units	Definition
UNCERT_LIM	Percent	Sets the uncertain limit for PCT_INCL. If PCT_INCL is below this limit, OUT receives the status, BAD.
UNIT_CONV	Scalar	User-specified numeric conversion factor to scale IN_2 units to IN_1 units. **Note** Do not include a time factor in the UNIT_CONV value. For example, when IN_1 is liters/hour and IN_2 is gallons/minute, UNIT_CONV = 3.785.

* These parameters are only visible when the function block is located in a Fieldbus device.

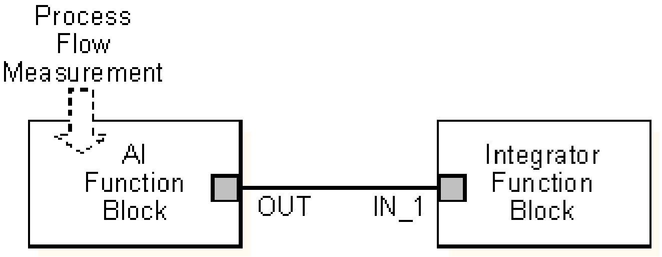
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Integrator Function Block

The Integrator function block is useful for calculating total flow, total mass, or volume over time. You can also use it to calculate total power, given the total energy.

Application Example: Flow Integration

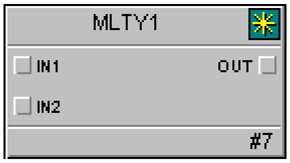
To totalize flow over a one-hour period, configure INTEG_TYPE = 0 to ? - periodic reset and configure CLOCK_PER to 3600. The following figure shows the function block diagram for this example:



Integrator Function Block Diagram Example

Multiply Function Block

The Multiply (MLTY) function block multiplies two to sixteen inputs and generates an output value. The block supports signal status propagation. There are no modes or alarm detection in the Multiply function block.



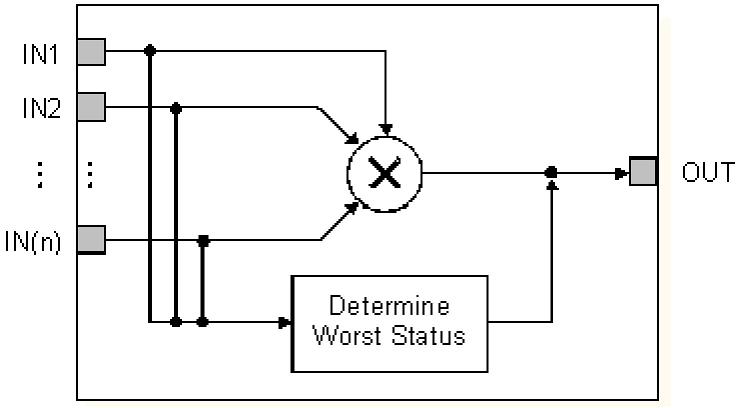
Multiply Function Block

IN1 through IN[n] are the input values and statuses to the block (as many as 16 inputs are allowed).

OUT is the output value and status.

Schematic Diagram - Multiply Function Block

The following figure shows the internal components of the Multiply function block:



Multiply Function Block Schematic Diagram

Block Execution - Multiply Function Block

The Multiply function block multiplies all input signals connected to the block and places the result at the output.

The number of inputs to the Multiply function block is an extensible parameter. The block default is two inputs. To add inputs, select the function block diagram, click the right mouse button, click Extensible Parameters, and modify the number of inputs. This creates additional input connectors for the block.

Status Handling - Multiply Function Block

The output status is set to the worst status among the inputs.

Parameters - Multiply Function Block

The following table lists the system parameters for the Multiply function block:

Multiply Function Block System Parameters

Parameter	Units	Description
IN1 to IN16	Determined by source	The analog input values and statuses. The number of inputs is an extensible parameter.
OUT	EU of IN	The analog output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Multiply Function Block

The Multiply function block is used as a mathematical operator and often is used with other Math function blocks. You can use the Multiply block to convert a signal from one type of engineering units to another.

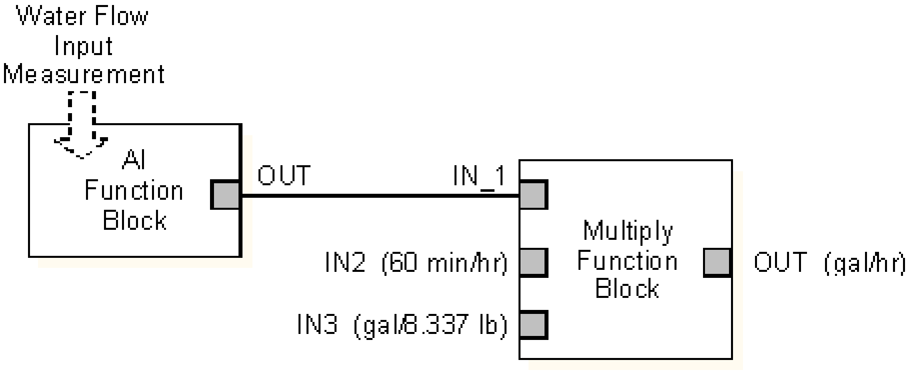
Application Example: Signal Conversion

Assume a transmitter measuring water flow is calibrated in pounds per minute, but the operator needs the information in gallons per hour. You can use the Multiply function block to calculate the correct value in the proper units.

For this example, you connect the input flow to IN1, set IN2 to 60.0 min/hr, and set IN3 to the specific volume of water (8.337 lb/gal). The block output reflects the multiplication of these values and units, as in the following equation:

$$\left(\frac{\text{lb}}{\text{min}}\right) \times \left(\frac{60\ \text{min}}{\text{hr}}\right) \times \left(\frac{\text{gal}}{8.337\ \text{lb}}\right) = \frac{\text{gal}}{\text{hr}}$$

For example, when the input flow (IN1) is 10 lb/min, the output value (OUT) equals 71.97 gal/hr. The following is the function block figure for this example:



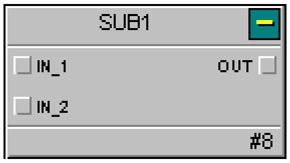
Multiply Function Block Diagram Example

Application Example: Applying a Gain Value

You can also use the Multiply function block to apply a gain value to a signal. In this case, you connect the signal to IN1 on the block and configure a gain value of 2.0. OUT reflects the signal value multiplied by the gain value.

Subtract Function Block

The Subtract (SUB) function block subtracts one input value from another input value and generates an output value. The block supports signal status propagation. There are no modes or alarm detection in the Subtract function block.



Subtract Function Block

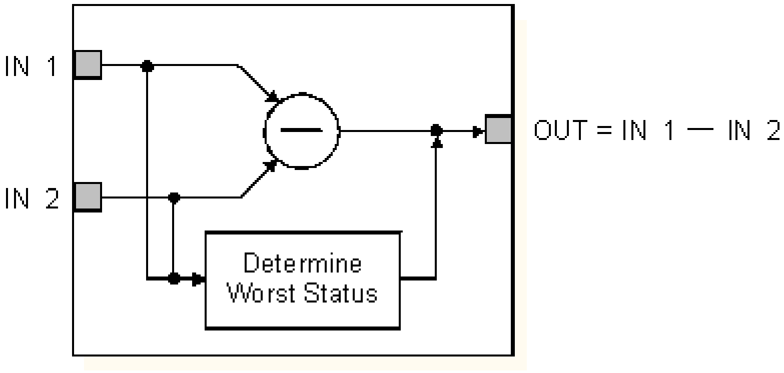
IN_1 is the first input value and status.

IN_2 is the second input value and status that is subtracted from the first input value.

OUT is the output value and status.

Schematic Diagram - Subtract Function Block

The following figure shows the internal components of the Subtract function block:



Subtract Function Block Schematic Diagram

Block Execution - Subtract Function Block

The Subtract function block accepts two signals (IN_1 and IN_2), takes the difference, and generates an output value. The following table shows Subtract function block outputs for different input values:

Divide Function Block Calculation Example

Parameter	Example 1	Example 2	Example 3
IN_1	6	74.37	5.5
IN_2	2	17.60	12.7
OUT	4	56.77	-7.2

Status Handling - Subtract Function Block

The output status is set to the worst status among the inputs.

Parameters - Subtract Function Block

The following table lists the system parameters for the Subtract function block:

Subtract Function Block System Parameters

Parameter	Units	Description
IN_1	Determined by source	The first input value and status.
IN_2	Determined by source	The second input value and status. This value is subtracted from IN_1.
OUT	Determined by IN_1 and IN_2	The output value and status.

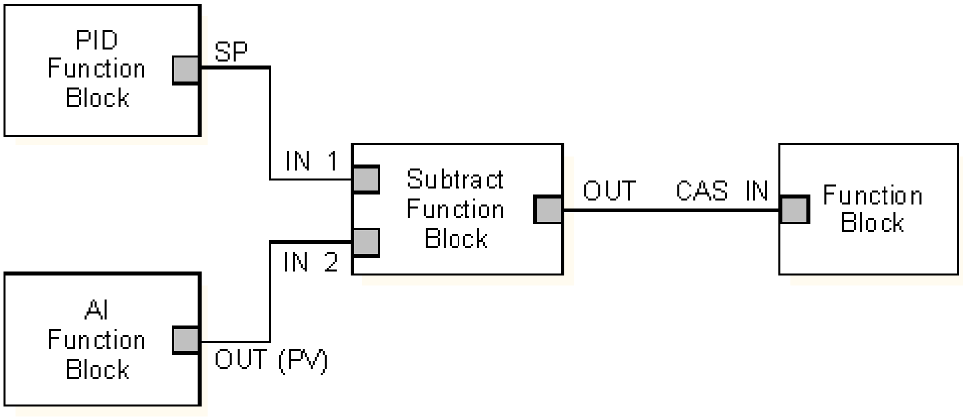
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Subtract Function Block

The Subtract function block is used as a mathematical operator and often is used with other Math function blocks. You can send the generated output to a controller or directly to the process.

Application Example: Calculate Offset

You can use the Subtract function block to determine the difference (error) between a setpoint and a process variable. In this case, the Subtract function block takes the setpoint (IN_1), subtracts the process variable (IN_2), and generates the error signal (OUT). You can send this value to another function block to calculate a cascade input.



Subtract Function Block Diagram Example

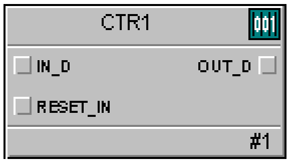
Timer/Counter Blocks

This chapter contains information on timer and counter function blocks in the DeltaV system.

Counter Function Block

The Counter (CTR) function block generates a discrete output value of True (1) when the count reaches a specified trip value. The block functions as an incremental (up) counter or a decremental (down) counter.

The Counter function block supports signal status propagation. There are no modes or alarm detection in the block.



Counter Function Block

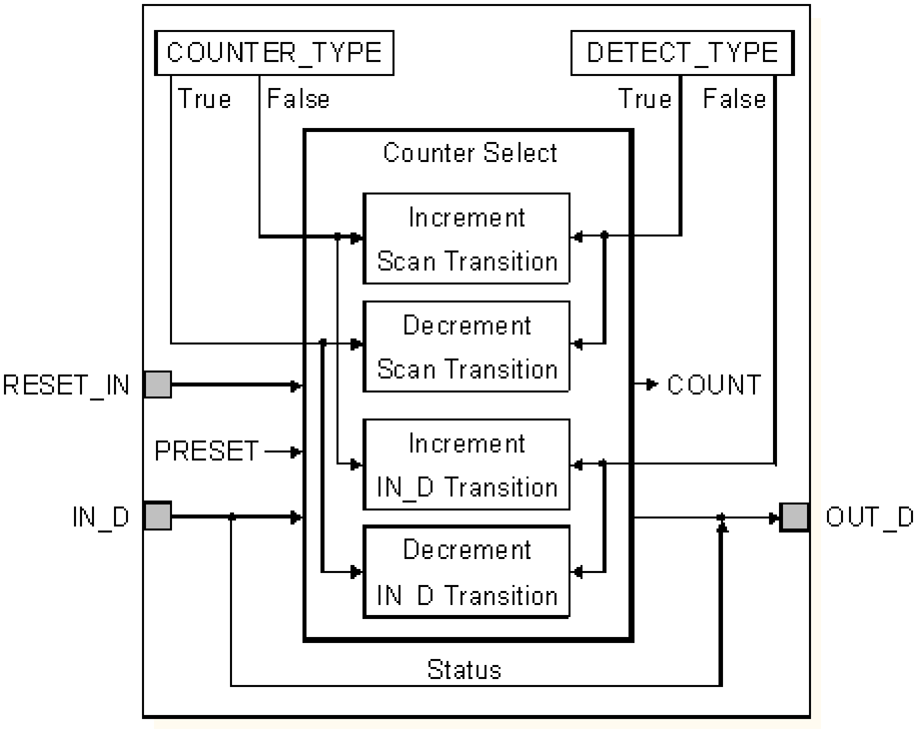
IN_D is the discrete input value that causes the counter to increment/decrement.

RESET_IN is the discrete input value that resets the counter.

OUT_D is the discrete output value and status.

Schematic Diagram - Counter Function Block

The following figure shows the internal components of the Counter function block:



Counter Function Block Schematic Diagram

Block Execution - Counter Function Block

You select the counter type by configuring the COUNTER_TYPE and DETECT_TYPE parameters. The COUNTER_TYPE parameter defines the type of counter (increment or decrement). The DETECT_TYPE parameter defines the transition count type.

When COUNTER_TYPE is Up Counter (False or 0):

- COUNT increases from zero based on IN_D.
- When COUNT is greater than or equal to the PRESET parameter value, OUT_D is set True and COUNT holds its value.
- When RESET_IN is True, COUNT is set to zero and OUT_D is set False.

When COUNTER_TYPE is Down Counter (True or 1):

- COUNT decreases from PRESET based on IN_D.
- When COUNT is less than or equal to zero, OUT_D is set True and COUNT holds its value.
- When RESET_IN is True, COUNT is set to PRESET and OUT_D is set False.

When DETECT_TYPE is Count on Rising Edge (False or 0):

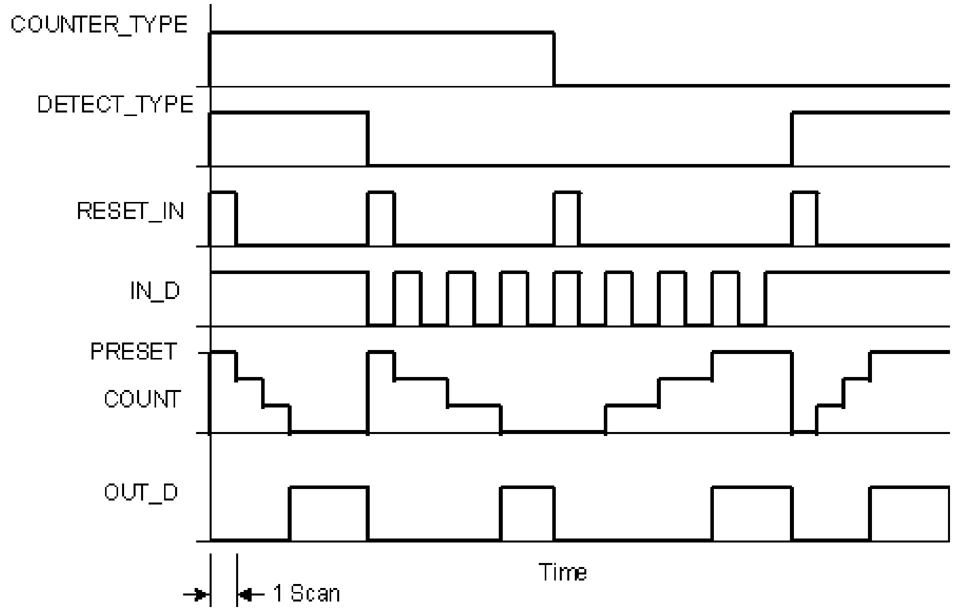
- The counter increments on upward transitions (False-to-True) of IN_D only.

When DETECT_TYPE is Count on TRUE (True or 1):

- The counter increments every scan that IN_D remains True. The counter value is represented in the COUNT parameter.

Each time COUNTER_TYPE or DETECT_TYPE is changed, the RESET_IN input is set True for one scan to reset the counter.

The following figure shows the timed response of the Counter function block for different input and parameter values:



Counter Function Block Timing Diagram

Status Handling - Counter Function Block

The output status is set to the input (IN_D) status. The status of RESET_IN does not affect the output status.

Parameters - Counter Function Block

The following table lists the system parameters for the Counter function block:

Counter Function Block System Parameters

Parameter	Units	Description
COUNT	None	The count value.
COUNTER_TYPE	None	Specifies the type of counter (False [0] = increment counter, True [1] = decrement counter).
DETECT_TYPE	None	Specifies the transition count type (False = counts when IN_D transitions from False to True [rising edge], True = counts when IN_D is True).
IN_D	None	The discrete input value and status.
OUT_D	None	The discrete output value and status. A True value indicates the counter has hit a trip condition.
PRESET	Seconds	The starting or trip value for the counter.
RESET_IN	None	Discrete input that resets the counter and sets OUT_D False.

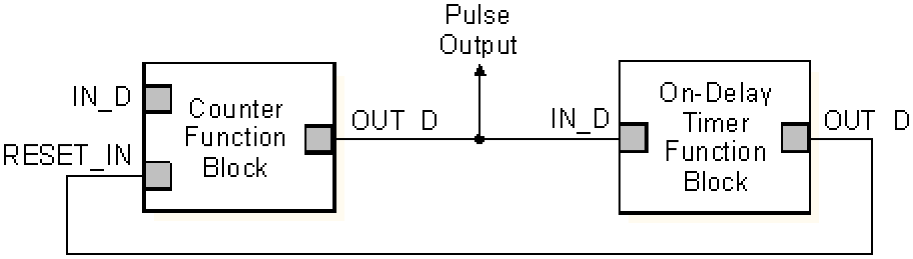
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Counter Function Block

You can use the Counter function block to trigger a polling event to gather system operating data.

Application Example: Trigger a Pulse Output

You can use the Counter function block with an On-Delay Timer function block to generate a pulse output. The following figure is the function block diagram for this application:



Counter Function Block Application Example

Date Time Event Function Block

The Date Time Event function block provides Time Of Day Timer functions and event timer functions. Use the Time Of Day functions to obtain date and time information in various formats. Use the event timer functions to schedule future events at a specified date and time.

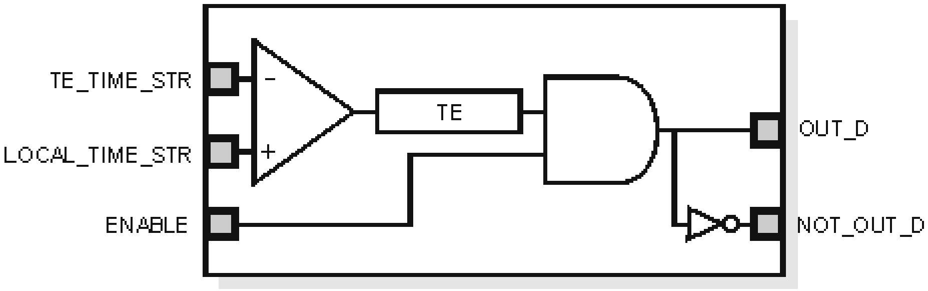
There are no modes or alarm detection in the block.

Date Time Event Function Block

OUT_D is the discrete output value and status.

Schematic Diagram - Date Time Event Function Block

The following figure shows the internal components of the Date Time Event function block:



Date Time Event Function Block Schematic Diagram

Block Execution - Date Time Event Function Block

For scheduling events, TE_TIME_STR and ENABLE are inputs. The block gets the internal (system) time (UTC_TIME_STR).

If RESET is true, the block returns its parameters to the default conditions (sets state to 1, or armed).

If STATE is idle, OUT_D is set to false. If this is the first time the block executes and if INTERVAL is not zero, the block sets the target event time.

If the TE_TIME_STR is still in the future the block sets STATE to armed.

During subsequent executions of the block it continues to compare the current time to TE_TIME_STR. When the two times are the same, OUT_D is set to true and STATE is set to terminal if there are more events scheduled or set to idle otherwise.

Daylight Savings Time Adjustments

The DTE function block has special behavior during Daylight Saving Time (DST) adjustment periods. In the case where a one-shot event is scheduled for a time that doesn't exist because of a forward DST adjustment, the event does not occur. In the case where a one-shot event is scheduled in the hour preceding a backwards DST adjustment, the second occurrence of the event does not occur. Periodic events are adjusted forward or backward, depending on the DST adjustment. Periodic events are not lost.

Relative Time versus Absolute Time

The DTE function block generates events based on Absolute Time (local time of day). If you use multiple DTE blocks to create a Relative Time dependency between events and the events straddle the DST adjustment period, the time relationship of the events changes according to the DST adjustment.

Status Handling - Date Time Event Function Block

The status of OUT_D and NOT_OUT_D are set to the status of the inputs.

Parameters - Date Time Event Function Block

The following table lists the system parameters for the Date Time Event function block:

Date Time Event Function Block System Parameters

Parameter	Units	Description

DIFF_TIME_STR	None	The time remaining before the scheduled event is activated, represented in ISO Period format:
		
		PDDDDDThh:mm:ss
		where:
		DDDDD = number of days
		hh = number of hours
		mm = number of minutes
		ss = number of seconds
		Set to P00000T00:00:00 for a one-time event. The minimum interval is P00000T00:00:05.

ENABLE	None	Used to either enable or disable the OUT_D parameter. (0=Disable, non-zero=Enable)
INTERVAL_STR	None	The time between automatically recurring scheduled events, represented in ISO Period format.
LOCAL_TIME_STR	None	The current Local Time represented as an ISO Time string.
NOT_OUT_D	None	Becomes false when the scheduled event has occurred and ENABLE is true. NOT_OUT_D has status.
OUT_D	None	Becomes true when the scheduled event has occurred and ENABLE is true. OUT_D has status.

Parameter	Units	Description
RESET	None	This value forces the function block to the IDLE state, when TRUE. RESET has status.
STATE	None	The internal value that indicates the current state of the function block (0=IDLE, 1=ARMED, 2=TERMINAL)
TE_TIME_STR	None	The time of day at which an event is scheduled, represented in local ISO Time format. This is known as *terminal event* or *TE* time.
UTC_TIME_STR	None	The current UTC Time represented as an ISO Time string.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Date Time Event Function Block

The Date Time Event function block can be used in several ways.

Example 1

Set the DTE Timer to activate its OUT_D parameter true every day at 5:00 PM local time, starting on September 14, 2002.

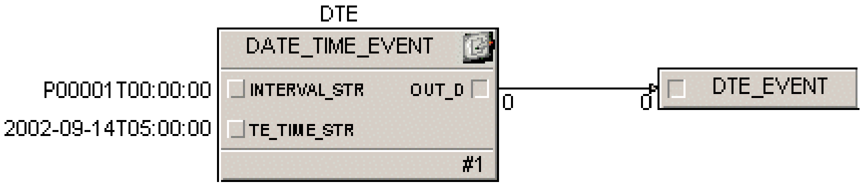
Expression usage example:

‘/TODTimer1.INTERVAL_STR’ := "P00001T00:00:00";

‘/TODTimer1.ENABLE’ := true;

‘/TODTimer1.TE_TIME_STR’ := "2002-09-14T05:00:00";

Function block usage example:



Example 2

Set a timer to output true every day at 5:00 PM local time for a duration of 60 seconds, starting on September 14, 2002.

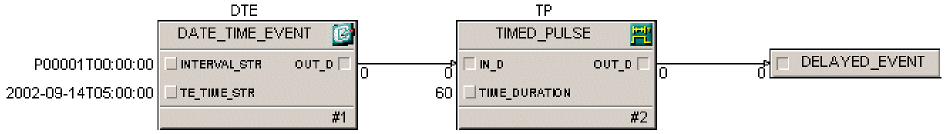
Using Expression Language:

‘/TODTimer1.INTERVAL_STR’ := "P00001T00:00:00";

‘/TODTimer1.ENABLE’ := true;

‘/TODTimer1.TE_TIME_STR := "2002-09-14T05:00:00";

Function block usage example:



Example 3

Set the DTE Timer to activate its OUT_D parameter true one time at 8:55 AM local time on December 9, 2002.

Expression usage example:

‘/TODTimer1.INTERVAL_STR’ := "P00000T00:00:00";

‘/TODTimer1.ENABLE’ := true;

‘/TODTimer1.TE_TIME_STR’ := "2001-12-09T08:55:00";

Example 4

Set the DTE Timer to activate its OUT_D parameter true one time at 8:55 AM UTC time on December 9, 2002. Also, use a module variable to display the time remaining before OUT_D will be activated.

Expression usage example:

‘/TODTimer1.INTERVAL_STR’ := "P00000T00:00:00";

‘/TODTimer1.ENABLE’ := true;

‘/DIFF_TIME_STR’ := ‘TODTimer1.DIFF_TIME_STR’;

‘/TODTimer1.TE_TIME_STR’ := "2001-12-09T08:55:00Z";

Example 5

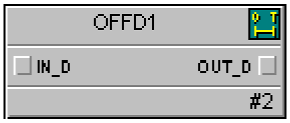
Use the DTE Timer to display the local time of day in a module variable.

Expression usage example:

‘/ISO_TIMESTAMP’ := ‘/TODTimer1.LOCAL_TIME_STR’;

Off-Delay Timer Function Block

The Off-Delay Timer (OFFD) function block delays the transfer of a False (0) discrete input value to the output by a specified time period. The block supports signal status propagation. There are no modes or alarm detection in the Off-Delay Timer function block.



Off-Delay Timer Function Block

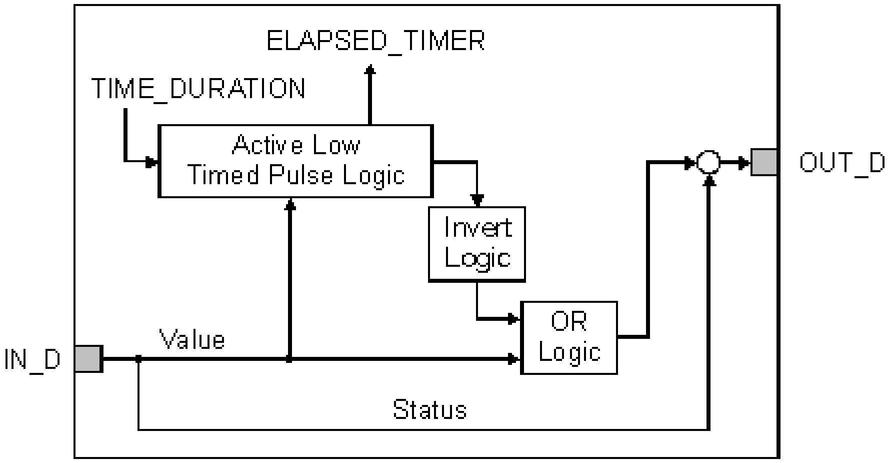
IN_D is the discrete input value and status used to trigger the timed discrete output value.

OUT_D is the discrete output value and status.

The Off-Delay Timer function block immediately transfers the discrete input value (IN_D) to the output (OUT_D) when IN_D is True (1). When IN_D transitions to False (0), OUT_D is reset to False after a specified time period (TIME_DURATION). During this time period, ELAPSED_TIMER tracks the time starting when IN_D transitions to False until the time specified by TIME_DURATION expires.

Schematic Diagram - Off-Delay Timer Function Block

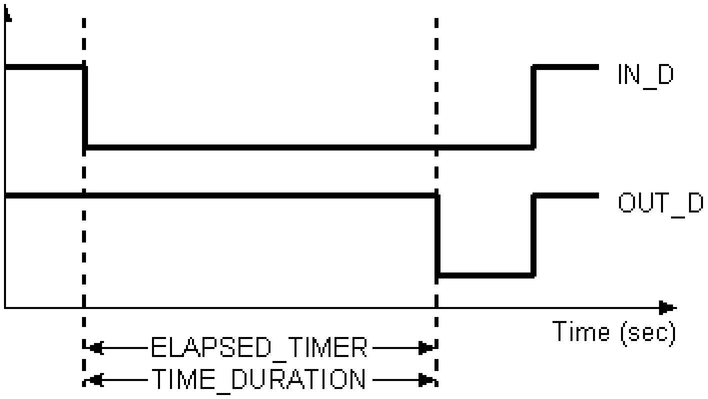
The following figure shows the internal components of the Off-Delay Timer function block:



Off-Delay Timer Function Block Schematic Diagram

Block Execution - Off-Delay Timer Function Block

The following figure shows the timed response of the Off-Delay Timer function block:



Off-Delay Timer Function Block Timing Diagram

When IN_D is True, OUT_D is set True and the elapsed time counter (ELAPSED_TIMER) is set to zero. When IN_D is False for longer than TIME_DURATION, OUT_D is set False.

Status Handling - Off-Delay Timer Function Block

The output status is set to the input status.

Parameters - Off-Delay Timer Function Block

The following table lists the system parameters for the Off-Delay Timer function block:

Off-Delay Timer Function Block System Parameters

Parameter	Units	Description
ELAPSED_TIMER	Seconds	The elapsed time since the False input value was active.
IN_D	None	The discrete input value and status used to trigger the timed discrete output value.
OUT_D	None	The discrete output value and status.
TIME_DURATION	Seconds	Specifies the time duration that must elapse before the False output value is applied.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Off-Delay Timer Function Block

You can use the Off-Delay Timer function block as a digital filter. For example, if input line power is susceptible to voltage drops, you can use the Off-Delay Timer function to prevent the output value from going low during temporary drops in voltage.

On-Delay Timer Function Block

The On-Delay Timer (OND) function block delays the transfer of a True (1) discrete input value to the output by a specified time period. The block supports signal status propagation. There are no modes or alarm detection in the On-Delay Timer function block.

On-Delay Timer Function Block

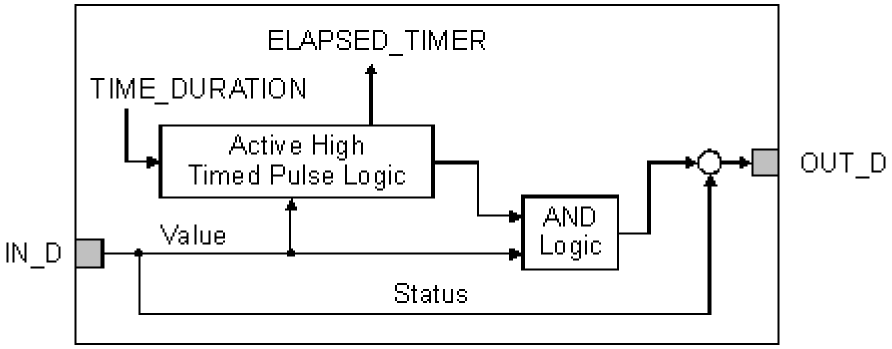
IN_D is the discrete input value and status used to trigger the timed discrete output value.

OUT_D is the discrete output signal and status.

The On-Delay Timer function block immediately transfers the discrete input value (IN_D) to OUT_D when IN_D is False. When IN_D transitions to True, OUT_D is set True after a configured time period (TIME_DURATION). During this time period, ELAPSED_TIMER tracks the time starting when IN_D transitions to True until the time specified by TIME_DURATION expires.

Schematic Diagram - On-Delay Timer Function Block

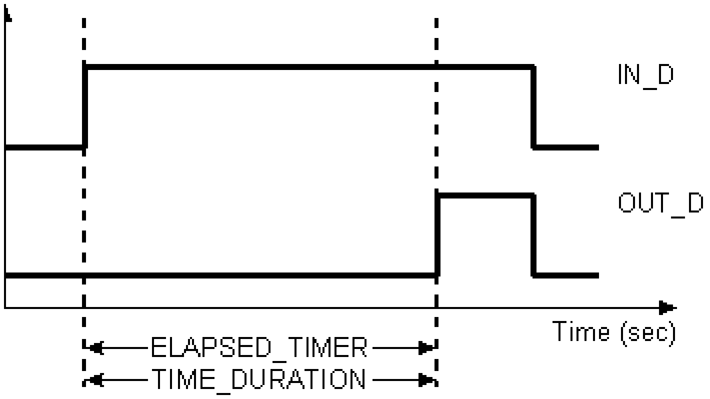
The following figure shows the internal components of the On-Delay Timer function block:



On-Delay Timer Function Block Schematic Diagram

Block Execution - On-Delay Timer Function Block

The following figure shows the timed response of the On-Delay Timer function block:



On-Delay Timer Function Block Timing Diagram

When IN_D is False, OUT_D is set False and the elapsed time counter (ELAPSED_TIMER) is set to zero. When IN_D is True longer than TIME_DURATION, OUT_D is set True.

Status Handling - On-Delay Timer Function Block

The output status is set to the input status.

Parameters - On-Delay Timer Function Block

The following table lists the system parameters for the On-Delay Timer function block:

On-Delay Timer Function Block System Parameters

Parameter	Units	Description
ELAPSED_TIMER	Seconds	The elapsed time since the True discrete input value was active.
IN_D	None	The discrete input value and status used to trigger the timed discrete output value.
OUT_D	None	The discrete output value and status.
TIME_DURATION	Seconds	Specifies the time duration that must elapse before the False discrete output value is applied.

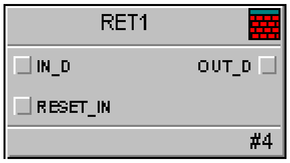
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - On-Delay Timer Function Block

You can use the On-Delay Timer function block as a digital filter. For example, if input line power is susceptible to voltage spikes, you can use the On-Delay Timer function to prevent the output value from going high when there is a temporary voltage spike in the input line.

Retentive Timer Function Block

The Retentive Timer (RET) function block generates a True (1) discrete output after the input has been True for a specified time period. The elapsed time the input has been True and the output value are reset when the reset input is set True.



Retentive Timer Function Block

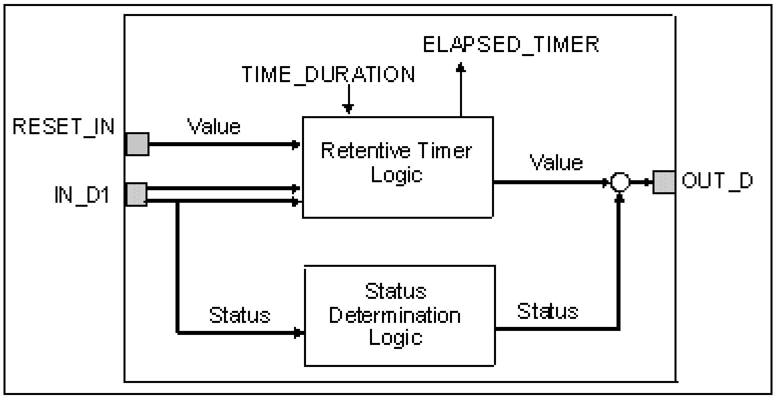
IN_D is the discrete input value and status to be timed.

RESET_IN is the discrete input value and status used to reset OUT_D and ELAPSED_TIMER.

OUT_D is the discrete output value and status.

Schematic Diagram - Retentive Timer Function Block

The following figure shows the internal components of the Retentive Timer function block:

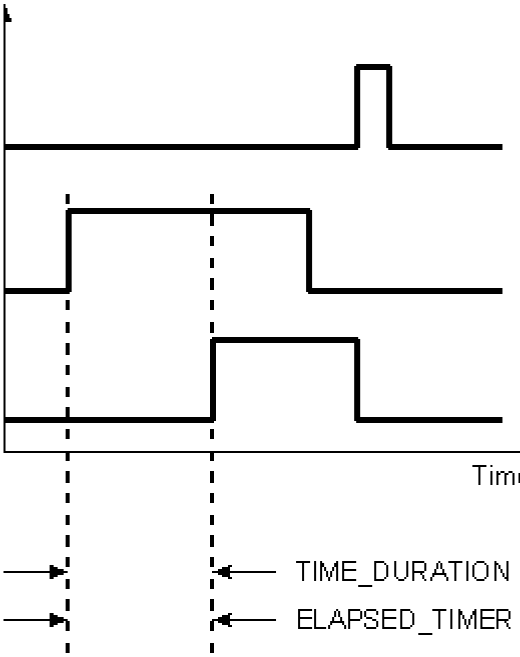


Retentive Timer Function Block Schematic Diagram

Block Execution - Retentive Timer Function Block

The block output (OUT_D) is set True when the input (IN_D) has been True for a specified time period (TIME_DURATION) while the RESET_IN input is False (0). When the RESET_IN input is False and the IN_D value transitions to False, the ELAPSED_TIMER stops and retains its value until IN_D transitions to True again. When the RESET_IN value transitions to True, the ELASPED_TIMER is reset to zero and OUT_D is set False.

The following figure shows the timed response of the Retentive Timer function block:



Retentive Timer Function Block Timing Diagram

Status Handling - Retentive Timer Function Block

The output status is set to Good: Process.

Parameters - Retentive Timer Function Block

The following table lists the system parameters for the Retentive Timer function block:

Retentive Timer Function Block System Parameters

Parameter	Units	Description
ELAPSED_TIMER	Seconds	The elapsed time IN_D has been True (1).
IN_D	None	The discrete input value and status.
OUT_D	None	The discrete output value and status.
RESET_IN	None	The reset discrete input value and status.
TIME_DURATION	Seconds	Specifies the time duration IN_D must be True before OUT_D is set True. This value must be greater than the scan rate of the parent module.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Retentive Timer Function Block

You can use the Retentive Timer function block to set an output to True when an oscillating input signal has been True for a specified time.

Timed Pulse Function Block

The Timed Pulse (TP) function block generates a True (1) discrete output for a specified time duration when the input makes a positive (False-to-True) transition. The output remains True even when the input returns to its initial discrete value and returns to its original False value only when the output is True longer than the specified time duration. Any 0 to True transition will cause a reset of the timer.

There are no modes or alarm detection in the block.

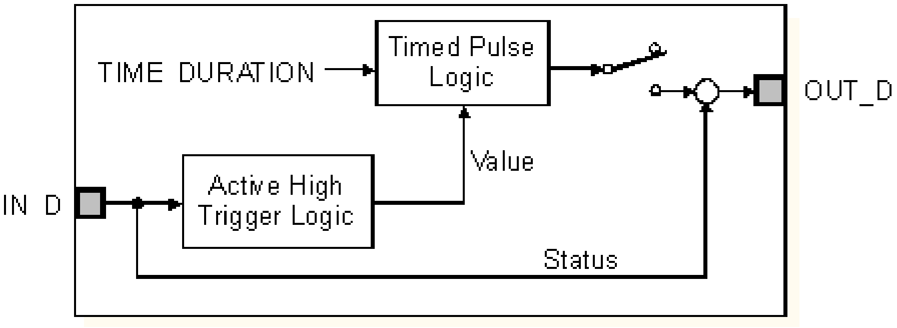
Timed Pulse Function Block

IN_D is the discrete input value and status used to trigger the timed discrete output value.

OUT_D is the discrete output value and status.

Schematic Diagram - Timed Pulse Function Block

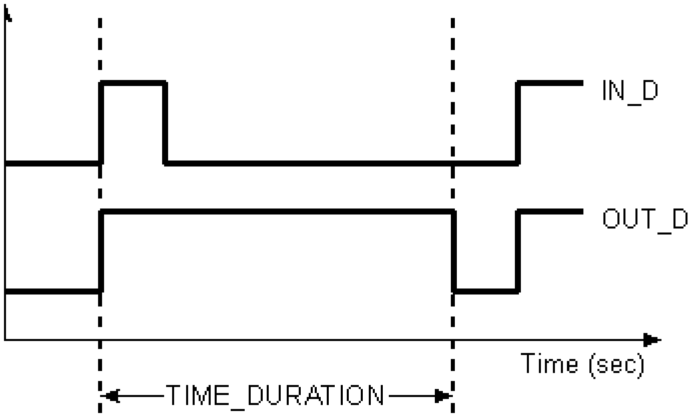
The following figure shows the internal components of the Timed Pulse function block:



Timed Pulse Function Block Schematic Diagram

Block Execution - Timed Pulse Function Block

The following figure shows the timed response of the Timed Pulse function block:



Timed Pulse Function Block Timing Diagram

Status Handling - Timed Pulse Function Block

The output status is set to Good: Process.

Parameters - Timed Pulse Function Block

The following table lists the system parameters for the Timed Pulse function block:

Timed Pulse Function Block System Parameters

Parameter	Units	Description
ELAPSED_TIMER	Seconds	The elapsed time since the input made a transition to True.
IN_D	None	The discrete input value and status used to trigger the timed discrete output value.
OUT_D	None	The timed discrete output value and status.
TIME_DURATION	Seconds	Specifies the time OUT_D remains True (1).

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Timed Pulse Function Block

The Timed Pulse function block sets the output True for a specified time. You can use the block to run a motor for a specified time period.

Energy Metering Blocks

This chapter contains information on energy metering function blocks in the DeltaV system.

Flow Metering (AGA_SI) Function Block

The AGA_SI function block is one of two flow metering function blocks on the Energy Metering palette. It is designed to calculate the flow of natural gas through orifice and turbine meters, but can be used for other gases and liquids (virtually any single phase, Newtonian fluid under turbulent flow).

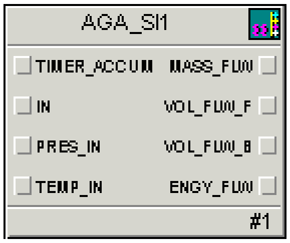
The AGA_SI function block is identical to the AGA_US block, except for the engineering units used. Each applicable parameter in the AGA_SI block has a particular SI engineering unit. If input or output parameters require other engineering units, conversion should be done upstream or downstream of the block. The other flow metering block, AGA_US, uses U.S. units.

The AGA_SI block calculates instantaneous mass flow, volumetric flow, and energy flow for natural gas using equations defined in AGA (American Gas Association) reports and ISO standards.

- Mass flow calculation for orifice metering per AGA3/1995 and ISO5167/1998
- Limits of use for orifice metering per AGA3/1995
- Calculation of densities and compressibility factors per AGA8/1994 (detail characterization method)
- Turbine metering calculations per AGA7/1996
- Calculation of gas heating value per ISO6976/1999 (real gas superior calorific value using *15/15* data)

The block calculates density for natural gases and other related hydrocarbon gases. For other fluids the density can be manually entered.

In addition to calculating instantaneous flow rates the block provides reset-able totalization parameters for base volumetric flow and energy flow.



IN is the differential pressure for an orifice meter in kPa (kilo Pascals) or volumetric flow from a PIN function block for a turbine meter in m3/hr (cubic meters per hour)

PRES_IN is the static pressure in kPa (kilo Pascals gage)

TEMP_IN is the fluid temperature in °C

TIMER_ACCUM is the input to reset totalization parameters typically wired from OUT_D of a DTE (Date Time Event) function block

MASS_FLW is the instantaneous mass flow rate in kg/hr (kilograms per hour)

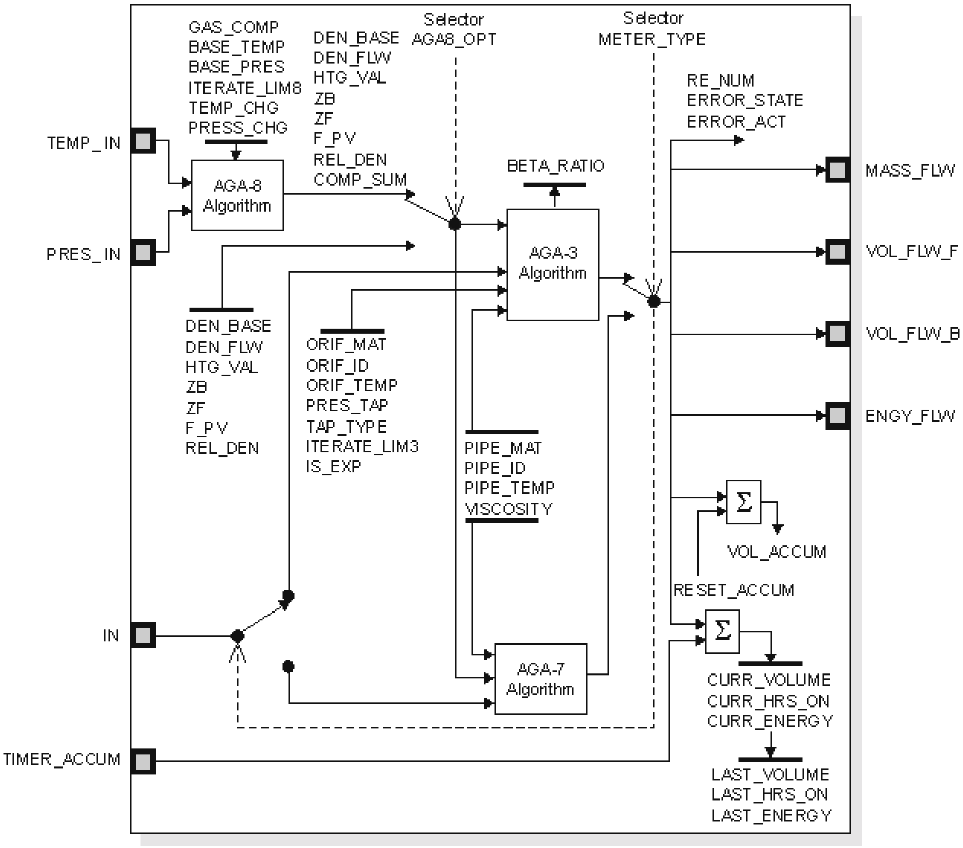
VOL_FLW_F is the volumetric flow rate at flowing conditions in m3/hr (cubic meters per hour)

VOL_FLW_B is the volumetric flow rate at base conditions in m3/hr (cubic meters per hour)

ENGY_FLW is the energy flow in GJ/hr (gigajoules per hour)

Schematic Diagram - AGA_SI Function Block

The following figure shows the internal components of the AGA function block:



AGA Function Block Schematic Diagram

Block Execution - AGA_SI Function Block

The configurer determines the algorithms used by the block by means of the following parameters:

METER_TYPE specifies whether an orifice (differential pressure) or turbine (pulse) meter is being used.

AGA8_OPT determines whether the AGA8 parameters are to be calculated by the block or manually entered. The AGA8 parameters include DEN_BASE and DEN_FLW (base and flowing density) and the report parameters ZB, ZF, F_PV, and REL_DEN (compressibility factors, supercompressibility, and relative density). When manual entry is chosen, the block does not calculate HTG_VAL (gas volumetric heating value), which otherwise is calculated using the ISO6976 technique, not AGA8.

The AGA8 parameters are calculated using the mole fractions for the 21 components in GAS_COMP and the base and flowing temperatures and pressures using the detail characterization method. The AGA8 parameters are calculated whenever PRES_IN or TEMP_IN changes from the last AGA8 calculation value by the amount specified in the tunable PRES_CHNG and TEMP_CHNG parameters (but at a minimum of every 60 seconds). The AGA8 parameters are calculated whenever GAS_COMP changes. If the fluid is not a natural gas or other hydrocarbon gas whose components are included in GAS_COMP, choose to manually enter the AGA8 parameters.

For orifice meters TAP_TYPE specifies the type of differential pressure tap, either Flange, Radius (D-D/2), or Corner taps. The block does not calculate flow rates for pipe taps (2.5D-8D).

PRES_TAP specifies the location of the static pressure tap, either Upstream or Downstream of the orifice plate.

The AGA3/ISO5167 algorithm calculates the mass flow rate when METER_TYPE is Differential Pressure. The coefficient of discharge is a function of pipe Reynolds number, which is a function of mass flow. Therefore the calculation of mass flow is an iterative one. The parameter ITERATE_LIM3 specifies the maximum number of iterations allowed. The default value is generally sufficient, but may need to be increased for viscous liquids. The entire flow equation is calculated each block execution, so averaging techniques are not necessary. From the mass flow the block uses the base and flowing densities from AGA8 to determine volumetric flow rates at base and flowing conditions. Like AGA3, the AGA8 algorithms are iterative. ITERATE_LIM8 specifies the maximum iterations allowed. If an iteration limit is violated, the condition is indicated in ERROR_STATE, as are any violations of the limits of use for orifice metering (see Status Handling).

When METER_TYPE is Turbine, the value on IN is the volumetric flow rate at flowing conditions (expected to be wired from the output of a Pulse Input function block). AGA_SI converts this flow to volumetric flow at base conditions and mass flow using the two density values.

Energy flow rate is determined from volumetric flow at base conditions and the volumetric heating value per ISO6976. In the AGA_SI block this is the superior heating value (not inferior) and is for the real gas (not ideal gas). It is based on the mole fractions in GAS_COMP and *15/15* data from ISO6976, which corresponds to a base temperature of 15 °C and base pressure of 0.101325 kPa.

Totalization Parameters

The block accumulates base volumetric flow each scan into two parameters, CURR_VOLUME (reset by TIMER_ACCUM) and VOL_ACCUM (reset by RESET_ACCUM). CURR_VOLUME is intended for daily or shift totals. VOL_ACCUM is intended for ad hoc totals to monitor short-term contract usage. The block accumulates energy flow in CURR_ENERGY and flow hours in CURR_HRS_ON. *Flow hours* is the number of hours of flow through the pipe where VOL_FLW_B is greater than zero.

The reset parameter TIMER_ACCUM is intended to be wired from OUT_D of a Date Time Event (DTE) function block, which allows resetting at regular intervals. When TIMER_ACCUM is 0, the block allows the accumulation of CURR_VOLUME, CURR_ENERGY, and CURR_HRS_ON. When greater than 0, the block copies the three CURR_ parameters to the three LAST_ parameters and resets the CURR_ parameters. *CURR* implies today or this shift or whatever the current period represents.

If the Restore parameter values after restart checkbox is selected in the module properties dialog, all of the accumulation parameters retain their current value during a module (partial) download and a controller restart. The accumulation parameters have status (see Status Handling)..

Status Handling - AGA_SI Function Block

The block determines the status of the output parameters and the internal accumulation parameters.

The status of the output parameters is a function of the status of the input parameters and the value of ERROR_STATE.

When ERROR_STATE is other than *Clear*, it impacts the status of output parameters. The error states include:

- **Gas Composition Sum Not 100%** - The sum of the mole percents in GAS_COMP does not equal 100.00 and AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. Contributes to Bad status on the output parameters.
- **Invalid Pipe or Orifice Size** – Actual beta ratio is less than 0.1 or greater than 0.75, or orifice ID is less than 11.43 mm, or pipe ID is less than 48.26 mm. Applicable when METER_TYPE is *Differential Pressure*. Contributes to Uncertain status on the output parameters.
- **Reynolds Number Out of Range** – Calculated pipe Reynolds number is less than 4000. Applicable when METER_TYPE is *Differential Pressure*. Contributes to Uncertain status on the output parameters.
- **AGA-8 Algo Not Convergent** – The required iterations exceeded ITERATE_LIM8, thus the AGA8 algorithm did not converge. Applicable when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. Contributes to Bad status on the output parameters (does not affect VOL_FLW_F when METER_TYPE is Turbine).
- **AGA-3 Algo Not Convergent** – The required iterations exceeded ITERATE_LIM3, thus the AGA3 algorithm did not converge. Contributes to Bad status on the output parameters.

The status of the output parameters is set to the worst of the status of the input parameters and the status based on ERROR_STATE.

The status of the accumulation parameters is Good until a value accumulated has Bad or Uncertain status. At that point the accumulation parameter has Uncertain status until it is reset.

There is a corresponding PCT_ parameter for each accumulation parameter. For example, CURR_VOLUME has PCT_CURR_VOLUME. The PCT_ parameter contains the percentage of the total in the corresponding parameter where the value being accumulated had Good status.

Parameters - AGA_SI Function Block

The following table lists the system parameters for the AGA function block:

AGA Block Parameters

Parameter	Units	Description
AGA8_OPT	N/A	Determines whether the block should calculate the AGA8 parameters using the detail method or they are to be manually entered. AGA8 parameters include DEN_FLW, DEN_BASE, ZF, ZB, F_PV, and REL_DEN. AGA8_OPT can be either: Calculate AGA-8 parameters using Detail Method - Select this option if the fluid is a natural gas. - Select this option if the fluid is a natural gas. Manually enter the AGA-8 parameters - Select this option for other single-phase, Newtonian fluids under turbulent flow. Manually enter DEN_FLW, DEN_BASE, VISCOSITY, and HTG_VAL (if applicable). Change HTG_VAL, ZF, ZB, F_PV, and REL_DEN to 0.0 when not applicable. - Select this option for other single-phase, Newtonian fluids under turbulent flow. Manually enter DEN_FLW, DEN_BASE, VISCOSITY, and HTG_VAL (if applicable). Change HTG_VAL, ZF, ZB, F_PV, and REL_DEN to 0.0 when not applicable.
BASE_PRES	kPa (kilo Pascals gage)	Base pressure, the reference pressure for base density and base volumetric flow.
BASE_TEMP	°C	Base temperature, the reference temperature for base density and base volumetric flow.
BETA_RATIO	N/A	Calculated ratio of orifice bore to pipe inside diameter after temperature correction. Has meaning when METER_TYPE is *Differential Pressure*.
COMP_SUM	None	The sum of the mole percents in GAS_COMP. If not between 99.995 and 100.005, results in a runtime ERROR_STATE of *Gas composition sum not 100%* if AGA8_Opt is *Calculate AGA-8 parameters using Detail Method*.
CURR_ENERGY	GJ (gigajoules)	The currently active accumulation of energy flow.
CURR_HRS_ON	Hours	The currently active accumulation of time with flow through the pipe (IN > LOW_CUT (differential pressure) or VOL_FLW_F > 0.0 (turbine)).
CURR_VOLUME	m3 (cubic meters)	The currently active accumulation of volumetric flow at base conditions.

Parameter	Units	Description
DEN_BASE	kg/m3 (kilograms per cubic meter)	Mass density at base conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
DEN_FLW	kg/m3 (kilograms per cubic meter)	Mass density at flowing conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
ENGY_FLW	GJ/hr (gigajoules per hour)	Instantaneous energy flow.
ERROR_ACT	N/A	Indicates that an error condition is active (0 = no error, 1 = an error condition is active). The error is defined as text in ERROR_STATE.
ERROR_STATE	N/A	The error state of the block. When other than Clear indicates the validity of the calculations is questionable: - Gas Composition Sum Not 100% - The sum of the mole percents in GAS_COMP does not equal 100.0 and AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. Results in Bad status on output parameters. - Invalid Pipe or Orifice Size – Actual beta ratio is less than 0.1 or greater than 0.75, or orifice ID is less than 11.43mm, or pipe ID is less than 48.26mm. Applicable when METER_TYPE is *Differential Pressure*. - Reynolds Number Out of Range – Calculated pipe Reynolds number is less than 4000. Applicable when METER_TYPE is *Differential Pressure*. - AGA-8 Algo Not Convergent – The required iterations exceeded ITERATE_LIM8, thus the AGA8 algorithm did not converge. Applicable when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. - AGA-3 Algo Not Convergent – The required iterations exceeded ITERATE_LIM3, thus the AGA3 algorithm did not converge.
F_PV	N/A	Supercompressibility factor, the square root of the ratio of the compressibility factors (base to flowing). Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.

Parameter	**Units**	**Description**
GAS_COMP	Mole fractions as percents	An array containing the mole fractions of 21 gas components. 1) Methane 2) Nitrogen 3) Carbon Dioxide 4) Ethane 5) Propane 6) Water 7) Hydrogen Sulfide 8) Hydrogen 9) Carbon Monoxide 10) Oxygen 11) i-Butane 12) n-Butane 13) i-Pentane 14) n-Pentane 15) n-Hexane 16) n-Heptane 17) n-Octane 18) n-Nonane 19) n-Decane 20) Helium 21) Argon
HTG_VAL	MJ/m3 (megajoules per cubic meter)	Volumetric heating value. Calculated per ISO6976 from the mole fractions in GAS_COMP, the superior calorific values of the components, and compressibility factors of the components (*15/15* data).
IN	kPa (kilo Pascals [differential pressure], m3/hr (cubic meters per hour [turbine])	When METER_TYPE is *Differential Pressure*, IN is the measured differential pressure. When METER_TYPE is *Turbine*, IN is the volumetric flow from a Pulse Input function block.
IS_EXP	N/A	Isentropic exponent of the natural gas or other fluid. Enter the value at expected flowing conditions. The default value is reasonable for natural gases. For an incompressible fluid enter –1.0. Not used when METER_TYPE is *Turbine*.
ITERATE_LIM3	N/A	The number of iterations limit for the AGA3 /ISO5167 mass flow calculation
ITERATE_LIM8	N/A	The number of iterations limit for the AGA8 density and compressibility factor calculations.
LAST_ENERGY	GJ (gigajoules)	The total accumulation of energy flow at the time of the last reset (using TIMER_ACCUM).
LAST_HRS_ON	Hours	The total accumulation of time with flow through the pipe at the time of the last reset (using TIMER_ACCUM.

Parameter	Units	Description
LAST_VOLUME	m3 (cubic meters)	The total accumulation of base volumetric flow at the time of the last reset (using TIMER_ACCUM).
LOW_CUT	kPa (kilo Pascals gage)	Value of IN (differential pressure) below which flow is assumed to be zero. Not applicable when METER_TYPE is *Turbine*.
MASS_FLW	kg/hr (kilograms per hour)	Instantaneous mass flow.
METER_TYPE	N/A	Meter type for flow measurement. The value can be *Differential Pressure* or *Turbine.*
ORIF_ID	mm (millimeters)	Orifice plate bore diameter at reference temperature ORIF_TEMP. Not applicable when METER_TYPE is *Turbine*.
ORIF_MAT	N/A	Orifice plate material. The value can be Stainless Steel, Monel, or Carbon Steel. Not applicable when METER_TYPE is *Turbine*.
ORIF_TEMP	°C	Reference temperature for the orifice plate. Not applicable when METER_TYPE is *Turbine.*
PCT_CURR_ENERGY	Percent	The percentage of the total in CURR_ENERGY where ENGY_FLW had Good status when the value was accumulated.
PCT_CURR_HRS_ON	Percent	The percentage of the total time in CURR_HRS_ON where VOL_FLW_F had Good status when the time was accumulated.
PCT_CURR_VOLUME	Percent	The percentage of the total in CURR_VOLUME where VOL_FLW_B had Good status when the value was accumulated.
PCT_VOL_ACCUM	Percent	The percentage of the total in VOL_ACCUM where VOL_FLW_B had Good status when the value was accumulated.
PCT_LAST_ENERGY	Percent	The percentage of the total in LAST_ENERGY where ENGY_FLW had Good status when the value was accumulated.
PCT_LAST_HRS_ON	Percent	The percentage of the total time in LAST_HRS_ON where VOL_FLW_F had Good status when the time was accumulated.
PCT_LAST_VOLUME	Percent	The percentage of the total in LAST_VOLUME where VOL_FLW_B had Good status when the value was accumulated.
PIPE_ID	mm (millimeters)	Pipe internal diameter at reference temperature PIPE_TEMP.

Parameter	Units	Description
PIPE_MAT	N/A	Pipe material. The value can be Stainless Steel, Monel, or Carbon Steel.
PIPE_TEMP	°C	Reference temperature for the pipe ID measurement.
PRES_CHNG	kPa (kilo Pascals)	The amount PRES_IN must differ from its value at the time of the previous AGA-8 calculation to trigger another AGA-8 calculation.
PRES_IN	kPa (kilo Pascals gage)	Measured static pressure
PRES_TAP	N/A	Location of the static pressure tap, either Upstream or Downstream of the orifice plate. Not applicable when METER_TYPE is *Turbine*.
RE_NUM	N/A	Calculated Pipe Reynolds number.
REL_DEN	N/A	Real gas relative density (specific gravity). Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
RESET_ACCUM	N/A	Resets VOL_ACCUM when greater than zero, then the block changes RESET_ACCUM back to zero.
TAP_TYPE	N/A	Orifice meter tap type, either Flange Taps, Radius Taps (D-D/2), or Corner Taps.
TEMP_CHNG	°C	How much TEMP_IN must change from the last AGA8 calculation temperature in order to trigger another AGA8 calculation.
TEMP_IN	°C	Measured temperature.
TIMER_ACCUM	N/A	When 0, allows the accumulation of CURR_VOLUME, CURR_ENERGY, and CURR_HRS_ON. When greater than 0, copies the three CURR_ parameters to the three LAST_ parameters and resets the CURR_ parameters. TIMER_ACCUM is typically wired from a Date Time Event function block.
VISCOSITY	kg/m-hr (kilograms per meter per hour)	The absolute viscosity of the natural gas or other fluid. Enter the value at expected flowing conditions based on laboratory analysis or other method.
VOL_ACCUM	m3 (cubic meters)	Accumulation of volumetric flow at base conditions. It is reset using RESET_ACCUM, which is expected to be done manually at irregular time intervals. Similar to CURR_VOLUME, which is also an accumulation of base volumetric flow, but CURR_VOLUME expected to be reset automatically at regular time intervals (using TIMER_ACCUM).

Parameter	Units	Description
VOL_FLW_B	m3/hr (cubic meters per hour)	Instantaneous volumetric flow at base conditions.
VOL_FLW_F	m3/hr (cubic meters per hour)	Instantaneous volumetric flow at flowing conditions.
ZB	N/A	Compressibility factor at base conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
ZF	N/A	Compressibility factor at flowing conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - AGA_SI Function Block

The primary considerations when applying the AGA_SI block are

- What type of meter is involved? Is it an orifice or turbine meter?
- Is the fluid a natural gas or a related hydrocarbon gas whose components are included in the array in the GAS_COMP parameter? Or is the fluid a liquid or gas whose components are not in GAS_COMP?

With an **orifice meter** choose *Differential Pressure* as the METER_TYPE and wire the IN input from an Analog Input function block connected to a differential pressure transmitter. The engineering units must be kPa (kilo Pascals gage).

With a **turbine meter** wire the IN input from a Pulse Input function block. OUT of the PIN block must in units of cubic meters per hour. Therefore the PULSE_VAL in the PIN block must be in units of cubic meters per pulse (inverse of the K-Factor) and TIME_UNITS must be Hours.

If the fluid is **natural gas** or related hydrocarbon gas, choose the default value for AGA8_OPT so that the block calculates the base and flowing densities of the gas per AGA8 and the report parameters (base and flowing compressibility factors, supercompressibility, and relative density). The block also calculates the volumetric heating value of the gas (per ISO6976).

If the fluid is a **liquid or other gas**, use AGA8_OPT to choose to manually enter these parameters. Typically you would enter the two densities and leave the other parameters at their default value. If the notion of energy is applicable, enter the volumetric heating value of the fluid at base conditions.

Runtime Limits for AGA8 Calculations

The AGA block does not check the input parameter limits for the AGA8 calculations at runtime. If any of the following parameters exceed the limits stated, calculations may not be accurate.

The limits are are exceeded if:

- BASE_TEMP less than 0 °C
- BASE_TEMP greater than 25 °C
- BASE_PRES less than 90 kPa
- BASE_PRES greater than 110 kPa
- DEN_BASE zero or negative
- TEMP_IN less than -130.0 °C
- TEMP_IN greater than 400.0 °C
- PRESS_IN greater than 280.0 kPa
- GAS_COMP
  - [1] (Methane) less than 45.0
  - [2] (Nitrogen) greater than 50.0
  - [3] (Carbon Dioxide) greater than 30.0
  - [4] (Ethane) greater than 10.0
  - [5] (Propane) greater than 4.0
  - [6] (Water) greater than 0.05
  - [7] (Hydrogen Sulfide) greater than 0.02
  - [8] (Hydrogen) greater than 10.0
  - [9] (Carbon Monoxide) greater than 3.0
  - [10] (Oxygen) greater than 21.0
  - [11] (i-Butane) + [12] (n-Butane) greater than 1.0
  - [13] (i-Pentane) + [14] (n-Pentane) greater than 0.3
  - [15] (Hexane) + [16] (Heptane) + [17] Octane + [18] Nonane + [19] (Decane) greater than 0.2
  - [20] (Helium) greater than 0.2
  - [21] (Argon) greater than 1.0

Other considerations

When you choose for the block to calculate the AGA8 parameters, configure the base temperature and pressure and enter the mole fractions of the components in GAS_COMP as percents. If you interface to an online gas chromatograph, use a Calc block expression to assign the value of the elements of GAS_COMP. For example, to assign the Methane mole percent

'^/AGA_SI1/GAS_COMP[1][1]' := '//GC01/R30001.CV'; where GC01 is the DST for a serial dataset

To assign the Nitrogen mole percent

'^/AGA_SI1/GAS_COMP[2][1]' := '//GC01/R30002.CV'; where GC01 is the DST for a serial dataset

If necessary, modify the value of one component before writing to GAS_COMP so that the sum of components is 100%. If the sum is not between 99.995 and 100.005, an error will occur in ERROR_STATE and the status of the outputs will be Bad.

Enter the absolute viscosity of the fluid at expected flowing conditions in the parameter VISCOSITY based on a laboratory analysis, calculation, or published data. The block does not calculate viscosity. The default value is reasonable for a natural gas. For orifice meters the value of viscosity impacts the calculated mass flow rate. For turbine meters viscosity is used only to calculate the pipe Reynolds number (which is also true for PIPE_ID, PIPE_MAT, and PIPE_TEMP).

For orifice meters enter a value for the isentropic exponent of the fluid in the parameter IS_EXP. This is the ratio of specific heats (constant pressure to constant volume). The default value is reasonable for natural gases. For an incompressible fluid enter a value of –1.0. The block does not use IS_EXP for turbine meters.

To reset the accumulation parameters CURR_VOLUME, CURR_ENERGY, and CURR_HRS_ON wire the TIMER_ACCUM input from the output of a Date Time Event function block. Configure the DTE block to produce a rising edge on its OUT_D parameter at the contract hour each day or at whatever interval you wish to reset the parameters. Configure the DTE block by setting INTERVAL_STR to the desired number of days, hours, minutes, or seconds between resets, for example P00001T00:00:00 for resetting daily or P00000T08:00:00 for resetting every eight hours. Then set TE_TIME_STR to the first time you want a reset to occur (in local time), for example, 2002-12-25T09:00:00 if the contract hour is 9 am. When TE_TIME_STR is a time in the past, the DTE block knows how to properly arm itself for the next reset upon any download.

Using AGA_SI with DeltaV Operate

DeltaV Operate has a dynamo set called AGA containing a group display faceplate for the AGA_SI (and AGA_US) block. The faceplate shows operating parameters for the block and is applicable for orifice and turbine meters (the IN parameter value is not visible when METER_TYPE is Turbine). In configure mode drag the dynamo onto a picture. Enter the path for the AGA_SI function block in the edit dialog.

Also included in the dynamo set is a header for the faceplate. The header includes the module tag and module description. Drag the dynamo onto the picture and position it above the faceplate. Enter the module path in the edit dialog.

Flow Metering (AGA_US) Function Block

The AGA_US function block is one of two flow metering function blocks on the Energy Metering palette. It is designed to calculate the flow of natural gas through orifice and turbine meters, but can be used for other gases and liquids (virtually any single phase, Newtonian fluid under turbulent flow).

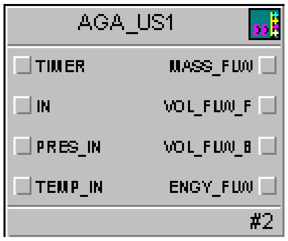
The AGA_US function block is identical to the AGA_SI block, except for the engineering units used. Each applicable parameter in the AGA_US block has a particular U.S. engineering unit. If input or output parameters require other engineering units, conversion should be done upstream or downstream of the block. The other flow metering block, AGA_SI, uses SI (System International) units.

The AGA_US block calculates instantaneous mass flow, volumetric flow, and energy flow for natural gas using equations defined in AGA (American Gas Association) reports and ISO standards.

- Mass flow calculation for orifice metering per AGA3/1995 and ISO5167/1998
- Limits of use for orifice metering per AGA3/1995
- Calculation of densities and compressibility factors per AGA8/1994 (detail characterization method)
- Turbine metering calculations per AGA7/1996
- Calculation of gas heating value per ISO6976/1999 (real gas superior calorific value using *15/15* data)

The block calculates density for natural gases and other related hydrocarbon gases. For other fluids the density can be manually entered.

In addition to calculating instantaneous flow rates the block provides reset-able totalization parameters for base volumetric flow and energy flow.



IN is the differential pressure (orifice meter) in inH2O at 68 °F or volumetric flow from a PIN function block (turbine meter) in ft3/hr

PRES_IN is the static pressure in psig

TEMP_IN is the fluid temperature in °F

TIMER_ACCUM is the input to reset totalization parameters typically wired from OUT_D of a DTE (Date Time Event) function block

MASS_FLW is the instantaneous mass flow rate in lb/hr

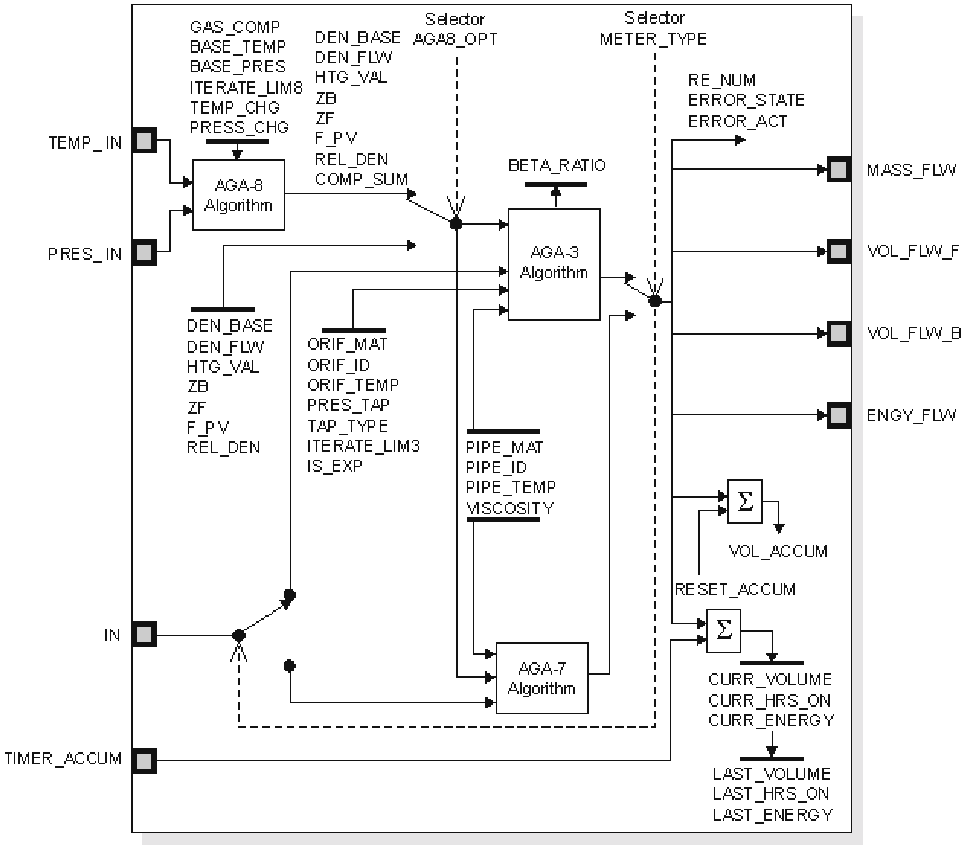
VOL_FLW_F is the volumetric flow rate at flowing conditions in ft3/hr

VOL_FLW_B is the volumetric flow rate at base conditions in ft3/hr

ENGY_FLW is the energy flow in MMBTU/hr

Schematic Diagram - AGA_US Function Block

The following figure shows the internal components of the AGA function block:



AGA Function Block Schematic Diagram

Block Execution - AGA_US Function Block

The configurer determines the algorithms used by the block by means of the following parameters:

METER_TYPE specifies whether an orifice (differential pressure) or turbine (pulse) meter is being used.

AGA8_OPT determines whether the AGA8 parameters are to be calculated by the block or manually entered. The AGA8 parameters include DEN_BASE and DEN_FLW (base and flowing density) and the report parameters ZB, ZF, F_PV, and REL_DEN (compressibility factors, supercompressibility, and relative density). When manual entry is chosen, the block does not calculate HTG_VAL (gas volumetric heating value), which otherwise is calculated using the ISO6976 technique, not AGA8.

The AGA8 parameters are calculated using the mole fractions for the 21 components in GAS_COMP and the base and flowing temperatures and pressures using the detail characterization method. The AGA8 parameters are calculated whenever PRES_IN or TEMP_IN changes from the last AGA8 calculation value by the amount specified in the tunable PRES_CHNG and TEMP_CHNG parameters (but at a minimum of every 60 seconds). The AGA8 parameters are calculated whenever GAS_COMP changes. If the fluid is not a natural gas or other hydrocarbon gas whose components are included in GAS_COMP, choose to manually enter the AGA8 parameters.

For orifice meters TAP_TYPE specifies the type of differential pressure tap, either Flange, Radius (D-D/2), or Corner taps. The block does not calculate flow rates for pipe taps (2.5D-8D).

PRES_TAP specifies the location of the static pressure tap, either Upstream or Downstream of the orifice plate.

The AGA3/ISO5167 algorithm calculates the mass flow rate when METER_TYPE is Differential Pressure. The coefficient of discharge is a function of pipe Reynolds number, which is a function of mass flow. Therefore the calculation of mass flow is an iterative one. The parameter ITERATE_LIM3 specifies the maximum number of iterations allowed. The default value is generally sufficient, but may need to be increased for viscous liquids. The entire flow equation is calculated each block execution, so averaging techniques are not necessary. From the mass flow the block uses the base and flowing densities from AGA8 to determine volumetric flow rates at base and flowing conditions. Like AGA3, the AGA8 algorithms are iterative. ITERATE_LIM8 specifies the maximum iterations allowed. If an iteration limit is violated, the condition is indicated in ERROR_STATE, as are any violations of the limits of use for orifice metering (see Status Handling).

When METER_TYPE is Turbine, the value on IN is the volumetric flow rate at flowing conditions (expected to be wired from the output of a Pulse Input function block). AGA_US converts this flow to volumetric flow at base conditions and mass flow using the two density values.

Energy flow rate is determined from volumetric flow at base conditions and the volumetric heating value per ISO6976. In the AGA_US block this is the superior heating value (not inferior) and is for the real gas (not ideal gas). It is based on the mole fractions in GAS_COMP and *15/15* data from ISO6976, which corresponds to a base temperature of 59 °F and base pressure of 14.696 psig.

Totalization Parameters

The block accumulates base volumetric flow each scan into two parameters, CURR_VOLUME (reset by TIMER_ACCUM) and VOL_ACCUM (reset by RESET_ACCUM). CURR_VOLUME is intended for daily or shift totals. VOL_ACCUM is intended for ad hoc totals to monitor short-term contract usage. The block accumulates energy flow in CURR_ENERGY and flow hours in CURR_HRS_ON. *Flow hours* is the number of hours of flow through the pipe where VOL_FLW_B is greater than zero.

The reset parameter TIMER_ACCUM is intended to be wired from OUT_D of a Date Time Event (DTE) function block, which allows resetting at regular intervals. When TIMER_ACCUM is 0, the block allows the accumulation of CURR_VOLUME, CURR_ENERGY, and CURR_HRS_ON. When greater than 0, the block copies the three CURR_ parameters to the three LAST_ parameters and resets the CURR_ parameters. *CURR* implies today or this shift or whatever the current period represents.

If the Restore parameter values after restart checkbox is selected in the module properties dialog, all of the accumulation parameters retain their current value during a module (partial) download and a controller restart. The accumulation parameters have status (see Status Handling)..

Status Handling - AGA_US Function Block

The block determines the status of the output parameters and the internal accumulation parameters.

The status of the output parameters is a function of the status of the input parameters and the value of ERROR_STATE.

When ERROR_STATE is other than *Clear*, it impacts the status of output parameters. The error states include:

- **Gas Composition Sum Not 100%** - The sum of the mole percents in GAS_COMP does not equal 100.00 and AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. Contributes to Bad status on the output parameters.
- **Invalid Pipe or Orifice Size** – Actual beta ratio is less than 0.1 or greater than 0.75, or orifice ID is less than 0.45 inches, or pipe ID is less than 1.9 inches. Applicable when METER_TYPE is *Differential Pressure*. Contributes to Uncertain status on the output parameters.
- **Reynolds Number Out of Range** – Calculated pipe Reynolds number is less than 4000. Applicable when METER_TYPE is *Differential Pressure*. Contributes to Uncertain status on the output parameters.
- **AGA-8 Algo Not Convergent** – The required iterations exceeded ITERATE_LIM8, thus the AGA8 algorithm did not converge. Applicable when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. Contributes to Bad status on the output parameters (does not affect VOL_FLW_F when METER_TYPE is Turbine).
- **AGA-3 Algo Not Convergent** – The required iterations exceeded ITERATE_LIM3, thus the AGA3 algorithm did not converge. Contributes to Bad status on the output parameters.

The status of the output parameters is set to the worst of the status of the input parameters and the status based on ERROR_STATE.

The status of the accumulation parameters is Good until a value accumulated has Bad or Uncertain status. At that point the accumulation parameter has Uncertain status until it is reset.

There is a corresponding PCT_ parameter for each accumulation parameter. For example, CURR_VOLUME has PCT_CURR_VOLUME. The PCT_ parameter contains the percentage of the total in the corresponding parameter where the value being accumulated had Good status.

Parameters - AGA_US Function Block

The following table lists the system parameters for the AGA function block:

AGA Block Parameters

Parameter	Units	Description
AGA8_OPT	N/A	Determines whether the block should calculate the AGA8 parameters using the detail method or they are to be manually entered. AGA8 parameters include DEN_FLW, DEN_BASE, ZF, ZB, F_PV, and REL_DEN. AGA8_OPT can be either: --Calculate AGA-8 parameters using Detail Method - Select this option if the fluid is a natural gas. - Select this option if the fluid is a natural gas. --Manually enter the AGA-8 parameters - Select this option for other single-phase, Newtonian fluids under turbulent flow. Manually enter DEN_FLW, DEN_BASE, VISCOSITY, and HTG_VAL (if applicable). Change HTG_VAL, ZF, ZB, F_PV, and REL_DEN to 0.0 when not applicable. - Select this option for other single-phase, Newtonian fluids under turbulent flow. Manually enter DEN_FLW, DEN_BASE, VISCOSITY, and HTG_VAL (if applicable). Change HTG_VAL, ZF, ZB, F_PV, and REL_DEN to 0.0 when not applicable.
BASE_PRES	psig (pounds per square inch gage)	Base pressure, the reference pressure for base density and base volumetric flow.
BASE_TEMP	°F	Base temperature, the reference temperature for base density and base volumetric flow.
BETA_RATIO	N/A	Calculated ratio of orifice bore to pipe inside diameter after temperature correction. Has meaning when METER_TYPE is *Differential Pressure*.
COMP_SUM	None	The sum of the mole percents in GAS_COMP. If not between 99.995 and 100.005, results in a runtime ERROR_STATE of *Gas composition sum not 100%* if AGA8_Opt is *Calculate AGA-8 parameters using Detail Method*.
CURR_ENERGY	MMBTU (million BTUs, decatherm)	The currently active accumulation of energy flow.
CURR_HRS_ON	Hours	The currently active accumulation of time with flow through the pipe (IN > LOW_CUT (differential pressure) or VOL_FLW_F > 0.0 (turbine)).
CURR_VOLUME	MCF (thousand cubic feet)	The currently active accumulation of volumetric flow at base conditions.

Parameter	**Units**	**Description**
DEN_BASE	lb/ft3 (pounts per cubic foot)	Mass density at base conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
DEN_FLW	lb/ft3 (pounds per cubic foot)	Mass density at flowing conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
ENGY_FLW	MMBTU/hr (million BTUs per hour)	Instantaneous energy flow.
ERROR_ACT	N/A	Indicates that an error condition is active (0 = no error, 1 = an error condition is active). The error is defined as text in ERROR_STATE.
ERROR_STATE	N/A	The error state of the block. When other than Clear indicates the validity of the calculations is questionable: - Gas Composition Sum Not 100% - The sum of the mole percents in GAS_COMP does not equal 100.0 and AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. - Invalid Pipe or Orifice Size – Actual beta ratio is less than 0.1 or greater than 0.75, or orifice ID is less than 0.45 inches, or pipe ID is less than 1.9 inches. Applicable when METER_TYPE is *Differential Pressure*. - Reynolds Number Out of Range – Calculated pipe Reynolds number is less than 4000. Applicable when METER_TYPE is *Differential Pressure*. - AGA-8 Algo Not Convergent – The required iterations exceeded ITERATE_LIM8, thus the AGA8 algorithm did not converge. Applicable when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*. - AGA-3 Algo Not Convergent – The required iterations exceeded ITERATE_LIM3, thus the AGA3 algorithm did not converge. .
F_PV	N/A	Supercompressibility factor, the square root of the ratio of the compressibility factors (base to flowing). Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.

Parameter	Units	Description
GAS_COMP	Mole fractions as percents	An array containing the mole fractions of 21 gas components. - Methane - Nitrogen - Carbon Dioxide - Ethane - Propane - Water - Hydrogen Sulfide - Hydrogen - Carbon Monoxide - Oxygen - i-Butane - n-Butane - i-Pentane - n-Pentane - n-Hexane - n-Heptane - n-Octane - n-Nonane - n-Decane - Helium - Argon
HTG_VAL	BTU/ft3 (BTUs per cubic foot)	Volumetric heating value. Calculated per ISO6976 from the mole fractions in GAS_COMP, the superior calorific values of the components, and compressibility factors of the components (*15/15* data).
IN	inH2O at 68 °F (inches of water) [differential pressure], ft3/hr (cubic feet per hour) [turbine]	When METER_TYPE is *Differential Pressure*, IN is the measured differential pressure. When METER_TYPE is *Turbine*, IN is the volumetric flow from a Pulse Input function block. Note: To convert differential pressure in psi to inH2O at 68 °F multiply by 27.730. To convert differential pressure from inH2O at 60 °F to inH2O at 68 °F multiply by 1.00083
IS_EXP	N/A	Isentropic exponent of the natural gas or other fluid. Enter the value at expected flowing conditions. The default value is reasonable for natural gases. For an incompressible fluid enter –1.0. Not used when METER_TYPE is *Turbine*.
ITERATE_LIM3	N/A	The number of iterations limit for the AGA3 /ISO5167 mass flow calculation.
ITERATE_LIM8	N/A	The number of iterations limit for the AGA8 density and compressibility factor calculations.
LAST_ENERGY	MMBTU (million BTUs, decatherm)	The total accumulation of energy flow at the time of the last reset (using TIMER_ACCUM).

Parameter	Units	Description
LAST_HRS_ON	Hours	The total accumulation of time with flow through the pipe at the time of the last reset (using TIMER_ACCUM.
LAST_VOLUME	MCF (thousand cubic feet)	The total accumulation of base volumetric flow at the time of the last reset (using TIMER_ACCUM).
LOW_CUT	inH2O at 68 °F	Value of IN (differential pressure) below which flow is assumed to be zero. Not applicable when METER_TYPE is *Turbine*.
MASS_FLW	lb/hr (pounds per hour)	Instantaneous mass flow.
METER_TYPE	N/A	Meter type for flow measurement. The value can be *Differential Pressure* or *Turbine.*
ORIF_ID	inches	Orifice plate bore diameter at reference temperature ORIF_TEMP. Not applicable when METER_TYPE is *Turbine*.
ORIF_MAT	N/A	Orifice plate material. The value can be Stainless Steel, Monel, or Carbon Steel. Not applicable when METER_TYPE is *Turbine*.
ORIF_TEMP	°F	Reference temperature for the orifice plate. Not applicable when METER_TYPE is *Turbine.*
PCT_CURR_ENERGY	Percent	The percentage of the total in CURR_ENERGY where ENGY_FLW had Good status when the value was accumulated.
PCT_CURR_HRS_ON	Percent	The percentage of the total time in CURR_HRS_ON where VOL_FLW_F had Good status when the time was accumulated.
PCT_CURR_VOLUME	Percent	The percentage of the total in CURR_VOLUME where VOL_FLW_B had Good status when the value was accumulated.
PCT_VOL_ACCUM	Percent	The percentage of the total in VOL_ACCUM where VOL_FLW_B had Good status when the value was accumulated.
PCT_LAST_ENERGY	Percent	The percentage of the total in LAST_ENERGY where ENGY_FLW had Good status when the value was accumulated.
PCT_LAST_HRS_ON	Percent	The percentage of the total time in LAST_HRS_ON where VOL_FLW_F had Good status when the time was accumulated.
PCT_LAST_VOLUME	Percent	The percentage of the total in LAST_VOLUME where VOL_FLW_B had Good status when the value was accumulated.

Parameter	Units	Description
PIPE_ID	inches	Pipe internal diameter at reference temperature PIPE_TEMP.
PIPE_MAT	N/A	Pipe material. The value can be Stainless Steel, Monel, or Carbon Steel.
PIPE_TEMP	°F	Reference temperature for the pipe ID measurement.
PRES_CHNG	psi (pounds per square inch)	The amount PRES_IN must differ from its value at the time of the previous AGA-8 calculation to trigger another AGA-8 calculation.
PRES_IN	psig (pounds per square inch gage)	Measured static pressure
PRES_TAP	N/A	Location of the static pressure tap, either Upstream or Downstream of the orifice plate. Not applicable when METER_TYPE is *Turbine*.
RE_NUM	N/A	Calculated Pipe Reynolds number.
REL_DEN	N/A	Real gas relative density (specific gravity). Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
RESET_ACCUM	N/A	Resets VOL_ACCUM when greater than zero, then the block changes RESET_ACCUM back to zero.
TAP_TYPE	N/A	Orifice meter tap type, either Flange Taps, Radius Taps (D-D/2), or Corner Taps.
TEMP_CHNG	°F	How much TEMP_IN must change from the last AGA8 calculation temperature in order to trigger another AGA8 calculation.
TEMP_IN	°F	Measured temperature.
TIMER_ACCUM	N/A	When 0, allows the accumulation of CURR_VOLUME, CURR_ENERGY, and CURR_HRS_ON. When greater than 0, copies the three CURR_ parameters to the three LAST_ parameters and resets the CURR_ parameters. TIMER_ACCUM is typically wired from a Date Time Event function block.
VISCOSITY	lb/ft-hr (pounds per foot per hour)	The absolute viscosity of the natural gas or other fluid. Enter the value at expected flowing conditions based on laboratory analysis or other method.

Parameter	Units	Description
VOL_ACCUM	MCF (thousand cubic feet)	Accumulation of volumetric flow at base conditions. It is reset using RESET_ACCUM, which is expected to be done manually at irregular time intervals. Similar to CURR_VOLUME, which is also an accumulation of base volumetric flow, but CURR_VOLUME is expected to be reset automatically at regular time intervals (using TIMER_ACCUM).
VOL_FLW_B	ft3/hr (cubic feet per hour)	Instantaneous volumetric flow at base conditions.
VOL_FLW_F	ft3/hr (cubic feet per hour)	Instantaneous volumetric flow at flowing conditions.
ZB	N/A	Compressibility factor at base conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.
ZF	N/A	Compressibility factor at flowing conditions. Calculated when AGA8_OPT is *Calculate AGA-8 parameters using Detail Method*, otherwise retains its previous value or the value as entered.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - AGA_US Function Block

The primary considerations when applying the AGA_US block are

1. What type of meter is involved? Is it an orifice or turbine meter?
2. Is the fluid a natural gas or a related hydrocarbon gas whose components are included in the array in the GAS_COMP parameter? Or is the fluid a liquid or gas whose components are not in GAS_COMP?

With an **orifice meter** choose *Differential Pressure* as the METER_TYPE and wire the IN input from an Analog Input function block connected to a differential pressure transmitter. The engineering units must be inH2O at 68 °F. Convert the AI block output prior to IN if necessary. To convert differential pressure in psi to inH2O at 68 °F multiply by 27.730. To convert differential pressure from inH2O at 60 °F to inH2O at 68 °F multiply by 1.00083.

With a **turbine meter** wire the IN input from a Pulse Input function block. OUT of the PIN block must in units of cubic feet per hour. Therefore the PULSE_VAL in the PIN block must be in units of cubic feet per pulse (inverse of the K-Factor) and TIME_UNITS must be Hours.

If the fluid is **natural gas** or related hydrocarbon gas, choose the default value for AGA8_OPT so that the block calculates the base and flowing densities of the gas per AGA8 and the report parameters (base and flowing compressibility factors, supercompressibility, and relative density). The block also calculates the volumetric heating value of the gas (per ISO6976).

If the fluid is a **liquid or other gas**, use AGA8_OPT to choose to manually enter these parameters. Typically you would enter the two densities and leave the other parameters at their default value. If the notion of energy is applicable, enter the volumetric heating value of the fluid at base conditions.

Runtime Limits for AGA8 Calculations

The AGA block does not check the input parameter limits for the AGA8 calculations at runtime. If any of the following parameters exceed the limits stated, calculations may not be accurate.

The limits are exceeded if:

- BASE_TEMP less than 32 °F
- BASE_TEMP greater than 77 °F
- BASE_PRES less than 13 psig
- BASE_PRES greater than 16 psig
- DEN_BASE zero or negative
- TEMP_IN less than -200.0 °F
- TEMP_IN greater than 760.0 °F
- PRESS_IN greater than 40000.0 psig
- GAS_COMP
  - [1] (Methane) less than 45.0
  - [2] (Nitrogen) greater than 50.0
  - [3] (Carbon Dioxide) greater than 30.0
  - [4] (Ethane) greater than 10.0
  - [5] (Propane) greater than 4.0
  - [6] (Water) greater than 0.05
  - [7] (Hydrogen Sulfide) greater than 0.02
  - [8] (Hydrogen) greater than 10.0
  - [9] (Carbon Monoxide) greater than 3.0
  - [10] (Oxygen) greater than 21.0
  - [11] (i-Butane) + [12] (n-Butane) greater than 1.0
  - [13] (i-Pentane) + [14] (n-Pentane) greater than 0.3
  - [15] (Hexane) + [16] (Heptane) + [17] Octane + [18] Nonane + [19] (Decane) greater than 0.2
  - [20] (Helium) greater than 0.2
  - [21] (Argon) greater than 1.0

Other considerations

When you choose for the block to calculate the AGA8 parameters, configure the base temperature and pressure and enter the mole fractions of the components in GAS_COMP as percents. If you interface to an online gas chromatograph, use a Calc block expression to assign the value of the elements of GAS_COMP. For example, to assign the Methane mole percent

'^/AGA_US1/GAS_COMP[1][1]' := '//GC01/R30001.CV'; where GC01 is the DST for a serial dataset^/AGA_US1/GAS_COMP[1][1]' := '//GC01/R30001.CV'; where GC01 is the DST for a serial dataset

To assign the Nitrogen mole percent

'^/AGA_US1/GAS_COMP[2][1]' := '//GC01/R30002.CV'; where GC01 is the DST for a serial dataset

If necessary, modify the value of one component before writing to GAS_COMP so that the sum of components is 100%. If the sum is not between 99.995 and 100.005, an error will occur in ERROR_STATE and the status of the outputs will be Bad.

Enter the absolute viscosity of the fluid at expected flowing conditions in the parameter VISCOSITY based on a laboratory analysis, calculation, or published data. The block does not calculate viscosity. The default value is reasonable for a natural gas. For orifice meters the value of viscosity impacts the calculated mass flow rate. For turbine meters viscosity is used only to calculate the pipe Reynolds number (which is also true for PIPE_ID, PIPE_MAT, and PIPE_TEMP).

For orifice meters enter a value for the isentropic exponent of the fluid in the parameter IS_EXP. This is the ratio of specific heats (constant pressure to constant volume). The default value is reasonable for natural gases. For an incompressible fluid enter a value of –1.0. The block does not use IS_EXP for turbine meters.

To reset the accumulation parameters CURR_VOLUME, CURR_ENERGY, and CURR_HRS_ON wire the TIMER_ACCUM input from the output of a Date Time Event function block. Configure the DTE block to produce a rising edge on its OUT_D parameter at the contract hour each day or at whatever interval you wish to reset the parameters. Configure the DTE block by setting INTERVAL_STR to the desired number of days, hours, minutes, or seconds between resets, for example, P00001T00:00:00 for resetting daily or P00000T08:00:00 for resetting every eight hours. Then set TE_TIME_STR to the first time you want a reset to occur (in local time), for example, 2002-12-25T09:00:00 if the contract hour is 9 am. When TE_TIME_STR is a time in the past, the DTE block knows how to properly arm itself for the next reset upon any download.

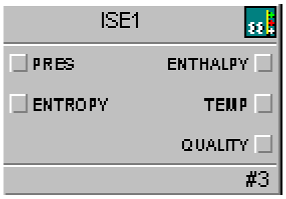
Using AGA_US with DeltaV Operate

DeltaV Operate has a dynamo set called AGA containing a group display faceplate for the AGA_US (and AGA_SI) block. The faceplate shows operating parameters for the block and is applicable for orifice and turbine meters (the IN parameter value is not visible when METER_TYPE is Turbine). In configure mode drag the dynamo onto a picture. Enter the path for the AGA_US function block in the edit dialog.

Also included in the dynamo set is a header for the faceplate. The header includes the module tag and module description. Drag the dynamo onto the picture and position it above the faceplate. Enter the module path in the edit dialog.

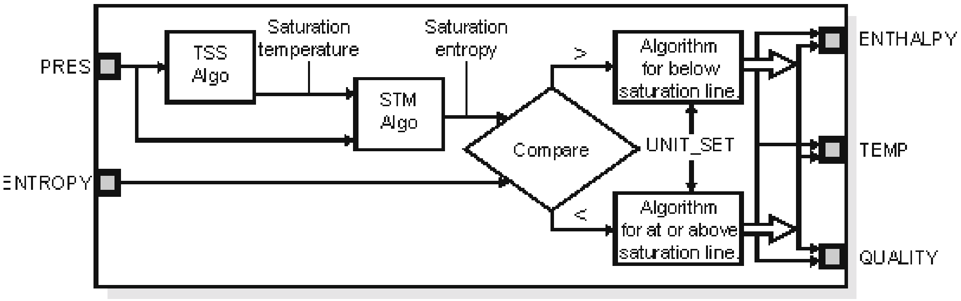
Isentropic Expansion (ISE) Function Block

The ISE function block calculates the final enthalpy for isentropic expansion of steam to a given pressure for given entropy. If the entropy-pressure point is in the two-phase region, the combined enthalpy of steam and water for constant entropy is provided. Steam quality (percent moisture) and steam temperature are also calculated. The ISE function block is valid for pressures -14 to 900 psig (-0.097 to 6.206 Mpa). The input entropy value should be between 0 and 2.6129 Btu/lb °R (10.940 kJ/kg K).



Schematic Diagram – Isentropic Expansion Function Block

The following figure shows the internal components of the ISE function block:



Isentropic Expansion Function Block Schematic Diagram

Status Handling – Isentropic Expansion Function Block

The status of the output parameters (ENTHALPY, TEMP, and QUALITY) is decided by the status of the input parameters (PRESS and ENTROPY). It is set to be the worst status of the input parameters. If the input PRES value is out of the range of –14 to 900 psig (-0.097 to 6.206 Mpa), the status is set to Uncertain. Also, the status of the output parameters is set to Bad and the output values are set to zero if the calculated result is not reliable due to the bad input value.

Parameters – Isentropic Expansion Function Block

The following table lists the system parameters for the ISE function block:

Isentropic Expansion Function Block Parameters

Parameter	Unit	Description
ENTHALPY	US: Btu/lb SI: kJ/kg	Enthalpy
ENTROPY	US: Btu/lb°R SI: kJ/kg K	Entropy
PRES	US: psig SI: MPa gage	Steam gauge pressure
QUALITY	Percent	Steam quality expressed in percent moisture
TEMP	US: °F SI: °C	Steam temperature
UNIT_SET	None	Defines the engineering units used in the block. The choices are United States Engineering Units (US) and System International Engineering Units (SI).

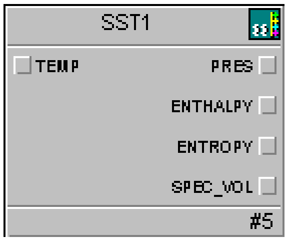
Application Information – Isentropic Expansion Function Block

The final enthalpy for isentropic expansion of steam to a given pressure is calculated for a given entropy. If the entropy-pressure point lies in the two phase region, the combined enthalpy of steam and water for constant entropy is provided. Steam quality (percent moisture) and steam temperature are also calculated.

The isentropic operation is valid for both the single-phase region and the two-phase region.

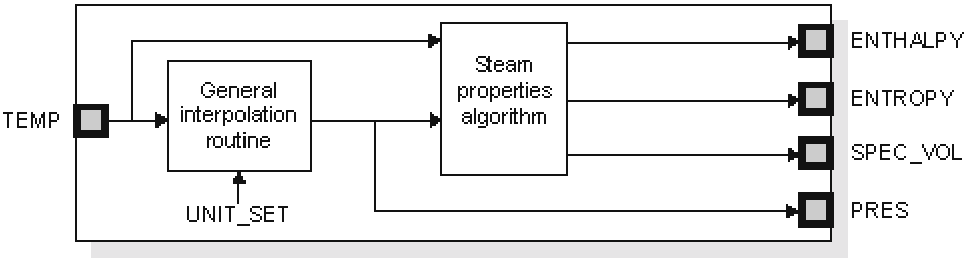
Saturated Steam Properties - Given Temperature (SST) Function Block

The SST function block calculates the steam enthalpy, entropy, specific volume, and pressure for saturation conditions specified by a given temperature. The block calculations apply to saturated steam. The calculation is limited to a temperature range of 35 °F to 700 °F (1.67 °C to 371.1 °C). An accuracy of 0.1% is obtained for temperature less than 540 °F (282.2 °C).



Schematic Diagram – Saturated Steam Properties - Given Temperature Function Block

The following figure shows the internal components of the SST function block.



Saturated Steam Properties - Given Temperature Function Block Schematic Diagram

Status Handling – Saturated Steam Properties - Given Temperature Function Block

The status of the output parameters (ENTHALPY, ENTROPY, SPEC_VOL, and PRES) is decided by the status of the input parameter (TEMP). It is set to be the worst status of the input parameters. If the input value is out of the range of 35 °F to 700 °F (1.67 °C to 371.1 °C), the status is set to Uncertain. Also, the status of the output parameters is set to Bad and the output values are set to zero if the input value is not a possible temperature for steam.

Parameters – Saturated Steam Properties - Given Temperature Function Block

The following table lists the system parameters for the SST function block:

Saturated Steam Properties - Given Temperature Function Block Parameters

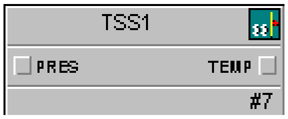
Parameter	Unit	Description
ENTHALPY	US: Btu/lb SI: kJ/kg	Enthalpy
ENTROPY	US: Btu/lb °R SI: kJ/kg K	Entropy
PRES	US: psig SI: MPa gage	Steam gauge pressure
SPEC_VOL	US: cu ft/lb SI: m3/kg	Specific volume
TEMP	US: °F SI: °C	Steam temperature
UNIT_SET	None	Defines the engineering units used in the block. The choices are United States Engineering Units (US) and System International Engineering Units (SI).

Application Information – Saturated Steam Properties - Given Temperature Function Block

The SST function block calculates the steam enthalpy, entropy, specific volume, and pressure for saturation conditions specified by a given temperature. The block calculations apply to saturated steam. The calculation is limited to a temperature range of 35 °F to 700 °F. An accuracy of 0.1% is obtained for temperature less than 540 °F.

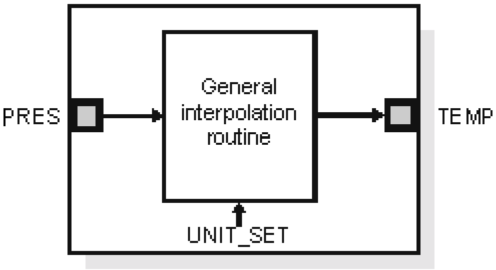
Saturated Temperature (TSS) Function Block

The TSS function block calculates the steam temperature at saturation at a given steam pressure. The block calculations apply to saturated steam. The calculation is limited to a pressure range of –14 to 2880 psig (-0.097 to 19.86 MPa).



Schematic Diagram – Saturated Temperature Function Block

The following figure shows the internal components of the TSS function block.



Saturated Temperature Function Block Schematic Diagram

Status Handling – Saturated Temperature Function Block

The status of the output parameter (TEMP) is decided by the status of the input parameter (PRES). It is set to be the status of the input parameter. If the input value is out of the range of –14 to 2880 psig (-0.097 to 19.86 MPa), the status is set to Uncertain. Also, the status of the output parameters will be set to Bad and the output values will be set to zero if the input value is far from reasonable.

Parameters – Saturated Temperature Function Block

The following table lists the system parameters for the TSS function block:

Saturated Temperature Function Block Parameters

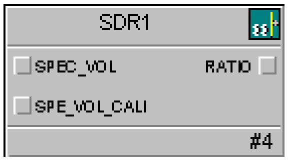
Parameter	Unit	Description
PRES	US: psig SI: MPa gage	Steam gauge pressure
TEMP	US: °F SI: °C	Steam temperature
UNIT_SET	None	Defines the engineering units used in the block. The choices are United States Engineering Units (US) and System International Engineering Units (SI).

Application Information – Saturated Temperature Function Block

The TSS function block calculates the steam temperature at saturation at a given steam pressure. The block calculations apply to saturated steam. The calculation is limited to a pressure range of –14 to 2880 psig (-0.097 to 19.86 MPa).

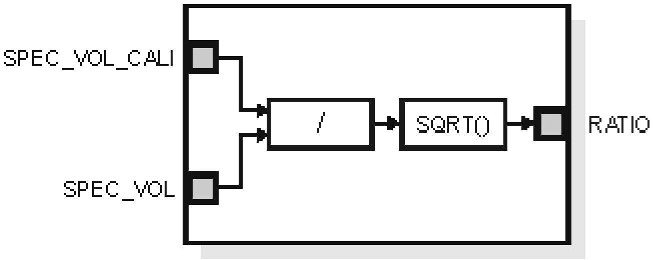
Steam Density Ratio (SDR) Function Block

The SDR function block calculates the square root of the ratio of steam density to the density of steam corresponding to a flow meter calibration pressure and temperature.



Schematic Diagram – Steam Density Ratio Function Block

The following figure shows the internal components of the SDR function block.



Steam Density Ratio Function Block Schematic Diagram

Status Handling – Steam Density Ratio Function Block

The status of the output parameter (RATIO) is decided by the status of the input parameters (SPE_VOL_CALI and SPEC_VOL). It is set to be the worst status of the input parameters. If the input value is less than zero, the status of the output parameter is set to Bad and the output value is set to zero.

Parameters – Steam Density Ratio Function Block

The following table lists the system parameters for the SDR function block:

Steam Density Ratio Function Block Parameters

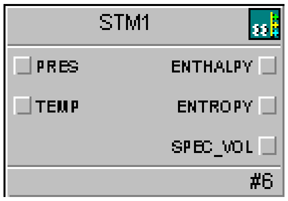
Parameter	Unit	Description
RATIO	None	Steam Density Ratio
SPE_VOL_CALI	US: cu ft/lb SI: m3/kg	Specific Volume at calibration condition
SPEC_VOL	US: cu ft/lb SI: m3/kg	Specific Volume

Application Information – Steam Density Ratio Function Block

The SDR function block calculates the square root of the ratio of steam density to the density of steam corresponding to a flow meter calibration pressure and temperature.

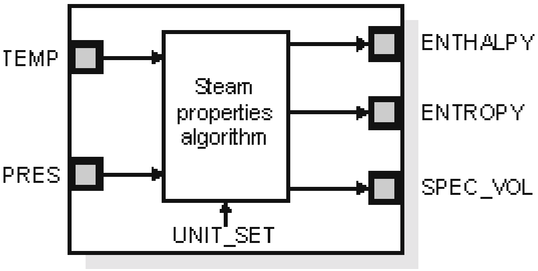
Steam Properties (STM) Function Block

The STM function block calculates steam enthalpy, entropy, and specific volume for a given gauge pressure and a given temperature. The block calculations apply to superheated steam and saturated steam. It covers the range 383°F to 860°F (195°C to 460°C). The calculation for superheated steam is generally accurate to 0.1%. Saturated steam properties are accurate to 0.1% for pressures less than 900 PSIG (6.206 MPa).



Schematic Diagram – Steam Properties Function Block

The following figure shows the internal components of the STM function block.



STM Function Block Schematic Diagram

Status Handling – Steam Properties Function Block

The status of the output parameters (ENTHALPY, ENTROPY, and SPEC_VOL) is decided by the status of the input parameters (PRES and TEMP). It is set to be the worst status of the input parameters. If the input values are not reasonable (that is, the pressure and temperature are not a possible combination for steam) and, as a result, the calculation result is less than zero, the status of the output parameters is set to Bad and the output values are set to zero. If the input temperature is outside the valid range, the block sets the output status to Uncertain. For example, the status is set to Uncertain if the input temperature is less than the saturation temperature for the input pressure.

Parameters – Steam Properties Function Block

The following table lists the system parameters for the STM function block:

Steam Properties Function Block Parameters

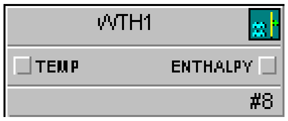
Parameter	Unit	Description
ENTHALPY	US: Btu/lb SI: kJ/kg	Enthalpy
ENTROPY	US: Btu/lb °R SI: kJ/kg K	Entropy
PRES	US: psig SI: MPa gage	Steam gauge pressure
SPEC_VOL	US: cu ft/lb SI: m3/kg	Specific volume
TEMP	US: °F SI: °C	Steam temperature
UNIT_SET	None	Defines the engineering units used in the block. The choices are United States Engineering Units (US) and System International Engineering Units (SI).

Application Information – Steam Properties Function Block

The STM function block calculates steam enthalpy, entropy, and specific volume for a given gauge pressure and a given temperature. The block calculations apply to superheated steam and saturated steam. It covers the range 383°F to 860°F, or 195°C to 460°C. The calculation for superheated steam is generally accurate to 0.1%. Saturated steam properties are accurate to 0.1% for pressures less than 900 PSIG, or 6.206 MPa.

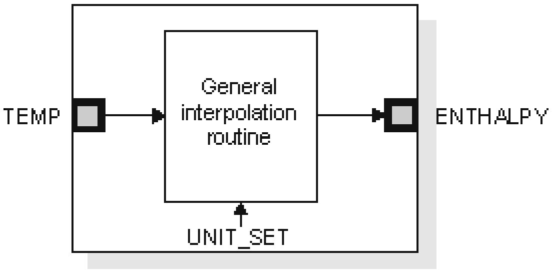
Water Enthalpy (WTH) Function Block

The WTH function block calculates the enthalpy of water for a specified temperature at saturation conditions. The calculation is limited to a temperature range of 32 °F to 704 °F (0 °C to 373.3 °C).



Schematic Diagram – Water Enthalpy Function Block

The following figure shows the internal components of the WTH function block.



WTH Function Block Schematic Diagram

Status Handling – Water Enthalpy Function Block

The status of the output parameter (ENTHALPY) is decided by the status of the input parameter (TEMP). It is set to be the status of the input parameter. If the input value is out of the range of 32 °F to 704 °F (0 °C to 373.3 °C), the status is set to Uncertain.

Parameters – Water Enthalpy Function Block

The following table lists the system parameters for the WTH function block:

Water Enthalpy Function Block Parameters

Parameter	Unit	Description
ENTHALPY	US: Btu/lb SI: kJ/kg	Water Enthalpy
TEMP	US: °F SI: °C	Water Temperature
UNIT_SET	None	Defines the engineering units used in the block. The choices are United States Engineering Units (US) and System International Engineering Units (SI).

Application Information – Water Enthalpy Function Block

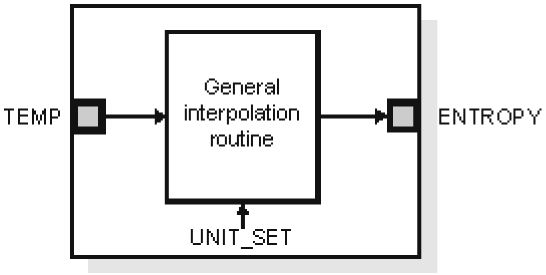
The WTH function block calculates the enthalpy of water for a specified temperature. The calculation is limited to a temperature range of 32 °F to 704 °F (0 °C to 373.3 °C).

Water Entropy (WTS) Function Block

The WTS function block calculates the entropy of water for a specified temperature at saturation conditions. The calculation is limited to a temperature range of 32 °F to 704 °F (0 °C to 373.3 °C).

Schematic Diagram – Water Entropy (WTS) Function Block

The following figure shows the internal components of the WTS function block.



Water Entropy Function Block Schematic Diagram

Status Handling – Water Entropy Function Block

The status of the output parameter (ENTROPY) is decided by the status of the input parameter (TEMP). It is set to be the status of the input parameter. If the input value is out of the range of 32 °F to 704 °F (0 °C to 373.3 °C), the status is set to Uncertain.

Parameters – Water Entropy Function Block

The following table lists the system parameters for the WTS function block:

Water Entropy Function Block Parameters

Parameter	Unit	Description
ENTROPY	US: Btu/lb °R SI: kJ/kg K	Water Entropy
TEMP	US: °F SI: °C	Water Temperature
UNIT_SET	None	Defines the engineering units used in the block. The choices are United States Engineering Units (US) and System International Engineering Units (SI).

Application Information – Water Entropy Function Block

The WTS function block calculates the entropy of water for a specified temperature. The calculation is limited to a temperature range of 32 °F to 704 °F (0 °C to 373.3 °C).

Logical Blocks

This chapter contains information on logical function blocks in the DeltaV system.

Action Function Block

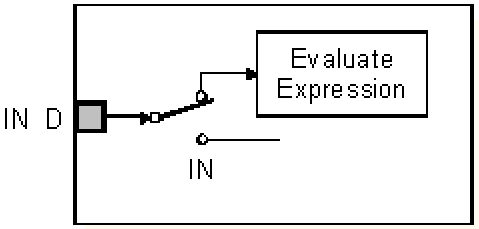
The Action (ACT) function block evaluates an expression when the input value is True. Mathematical functions, logical operators, and constants can be used in the expression. There are no modes, alarm detection, or status handling in the Action function block.

Action Function Block

IN_D is the discrete input value and status that initiates expression evaluation.

Schematic Diagram - Action Function Block

The following figure shows the internal components of the Action function block:



Action Function Block Schematic Diagram

Block Execution - Action Function Block

The Action function block evaluates an expression when the input (IN_D) is set True (1). The result is sent to an external reference parameter; there are no outputs in this block.

Expressions are structured text in a specific syntax and are made up of operands, operators, functions, constants, and keywords. You write expressions using the Expression Editor.

Refer to the Expressions topic for syntax guidelines for writing expressions and for information on the supported operands, operators, functions, constants, and keywords.

Status Handling - Action Function Block

The status of IN_D does not impact the evaluation of the expression. The expression is evaluated whenever IN_D is True (1).

Parameters - Action Function Block

The following table lists the system parameters for the Action function block:

Action Function Block System Parameters

Parameter	Units	Description
ALGO_OPTS	None	Algorithm options. When selected, the expression algorithm will abort after a read error. The Algorithm option is AbortOnReadErrors.
IN_D	None	The discrete input value and status.

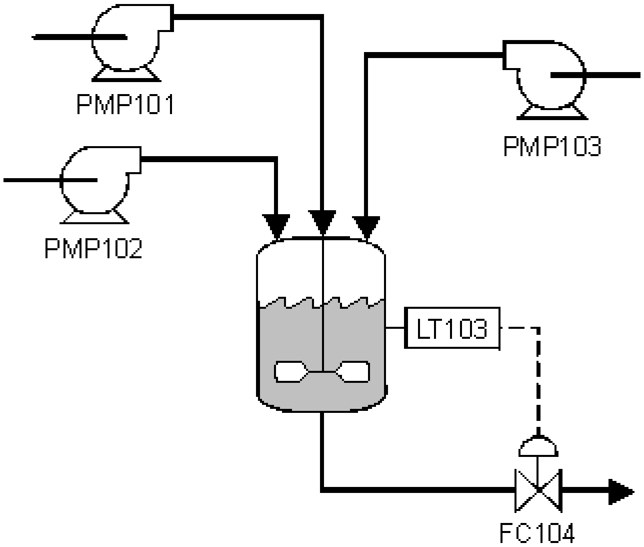
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Action Function Block

You can use the Action function block to implement complex calculations, signal conditioning, or functionality that is not available in the standard function blocks. You can use trigonometric, exponential, and power functions, as well as standard mathematical operations.

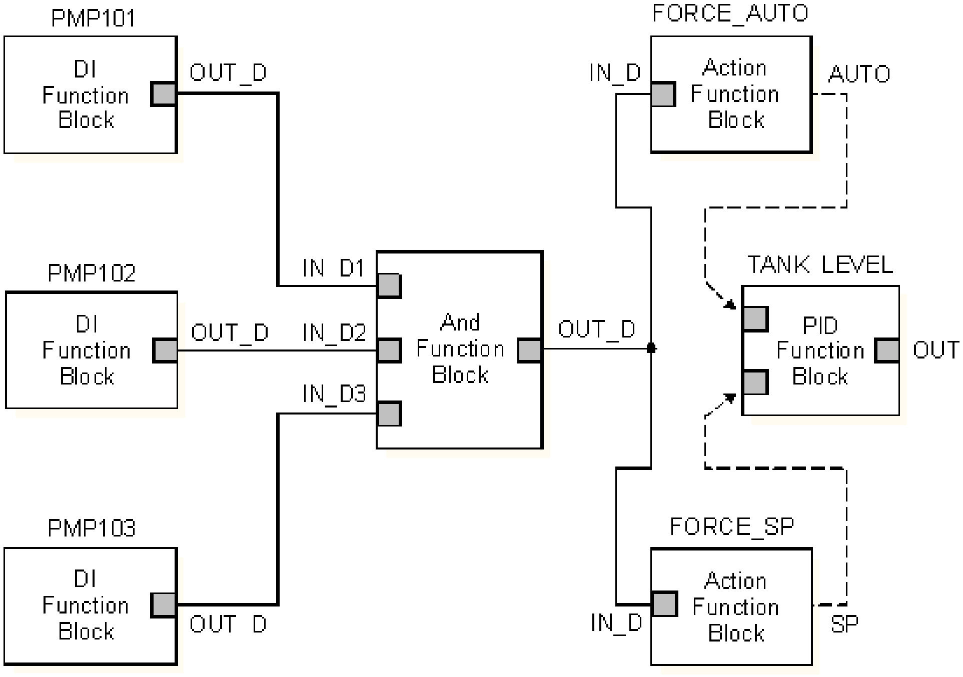
Application Example - Force Auto Control

Assume you have a tank that can be filled by up to three pumps with different ingredients. The following figure shows an example of this process:



Action Function Block Application Example

Assume you want the operator to be able to choose to run the tank level controller in Auto or Man mode when one or two ingredients are being pumped. However, when three ingredients are being pumped, you want the tank level controller to be forced to Auto mode at a predefined setpoint of 95%. You use two Action function blocks to enforce these requirements, as shown in the following function block diagram:



Action Function Block Diagram Example

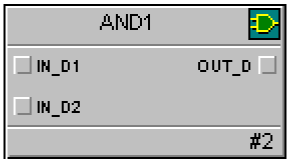
When all three pumps are on, the output of the And function block is True (1). This triggers the two Action blocks to execute their expressions. You configure the following expression in the FORCE_AUTO Action block to set the TANK_LEVEL PID controller target mode to Auto:

`'/TANK_LEVEL/MODE.TARGET' := AUTO;`You configure the following expression in the FORCE_SP Action block to set the TANK_LEVEL PID controller setpoint to 95%:

`'/TANK_LEVEL/SP.CV' := 95.0;`The Action function block expressions continue to execute as long as all three pumps are running and any operator attempt to change the TANK_LEVEL controller mode or setpoint is overridden.

And Function Block

The And function block generates a discrete output value based on the logical AND of two to sixteen discrete inputs. The block supports signal status propagation. There are no modes or alarm detection in the And function block.



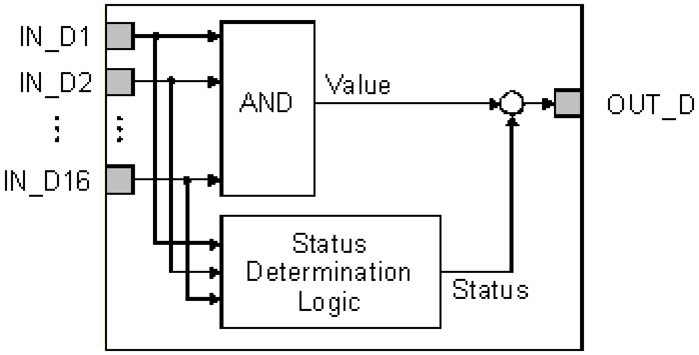
And Function Block

IN_D1 through IN_D[n] are the discrete input values and statuses (as many as 16 inputs).

OUT_D is the discrete output value and status.

Schematic Diagram - And Function Block

The following figure shows the internal components of the And function block:



And Function Block Schematic Diagram

Block Execution - And Function Block

The number of inputs to the And function block is an extensible parameter. The block default is two inputs. You add inputs by selecting the function block diagram, clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs. This creates additional input connectors for the block.

The And function block examines the inputs you define and applies the logical AND function to the inputs. When all inputs are True (1), the output is True. When one or more of the inputs is False (0), the output is False.

Status Handling - And Function Block

The output status is set to the worst status among the selected inputs unless at least one input is False and its status is not Bad. When this is the case, the output status is set to Good: Process.

Parameters - And Function Block

The following table lists the system parameters for the And function block:

And Function Block System Parameters

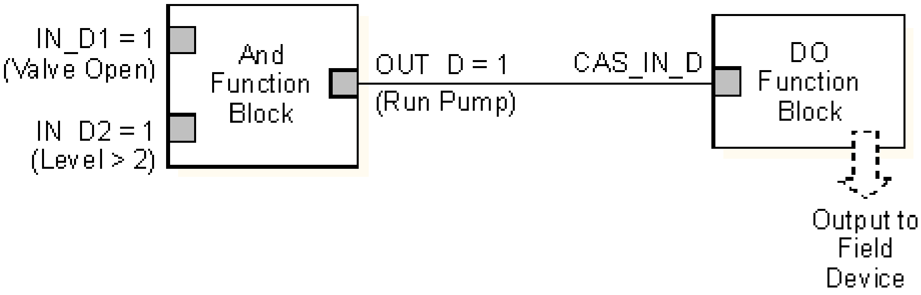
Parameter	Units	Description
IN_D1 to IN_D16	None	The discrete input values and statuses. The number of discrete inputs is an extensible parameter.
OUT_D	None	The discrete output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - And Function Block

The And function block is used to determine if all the selected discrete inputs are True. You can use the And block to stop a process in an emergency when one or more emergency stop conditions are met.

You can also use the And function block for an interlock condition to be sure a pump runs only when the feed valve is open and the level in the feed tank is above a minimum value. The valve and level transmitter values can be inputs to the And function block. The And block would send a signal to run the pump only when the two requirements were met. The resulting signal could be sent to a Discrete Output (DO) function block for additional processing, as in the following example:

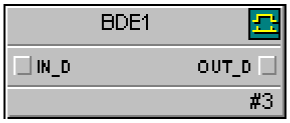


And Function Block Application Example

Bi-directional Edge (BDE) Trigger Function Block

The Bi-directional Edge Trigger (BDE) function block generates a True (1) discrete pulse output when the discrete input makes a positive (False-to-True) or a negative (True-to-False) transition since the last execution of the block. If there has been no transition, the discrete output of the block is False (0).

The Bi-directional Edge Trigger function block supports signal status propagation. There are no modes or alarm detection in the block.



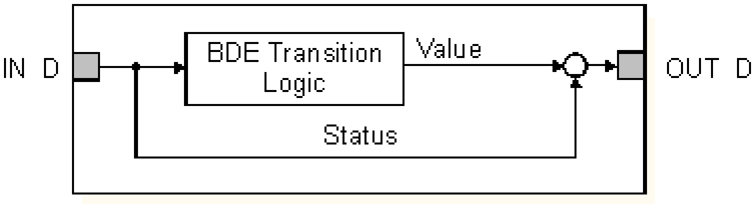
Bi-directional Trigger Function Block

IN_D is the discrete input signal and status.

OUT_D is the discrete output signal and status.

Schematic Diagram - Bi-directional Edge Trigger Function Block

The following figure shows the internal components of the Bi-directional Edge Trigger function block:

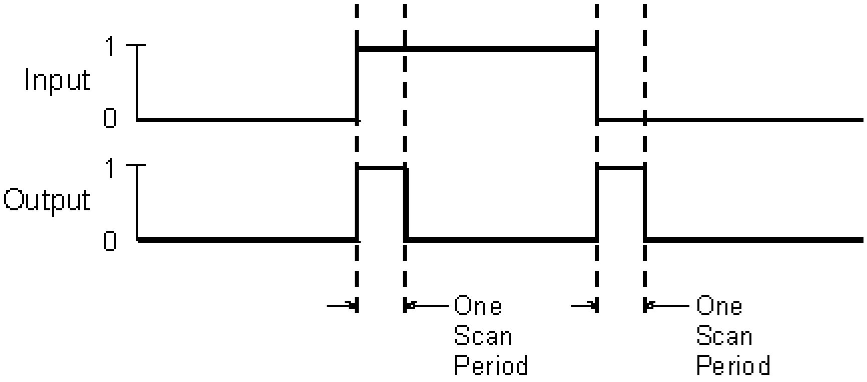


Bi-directional Edge Trigger Function Block Schematic Diagram

Block Execution - Bi-directional Edge Trigger Function Block

The Bi-directional Edge Trigger function block examines the input value, compares it to the previous input value, and sets the output True for one scan period when the input has changed. Otherwise, the output is False. The status of the output value is set to the status of the input value.

The following figure illustrates how the Bi-directional Edge Trigger function block responds to a change in input:



Bi-directional Edge Trigger Function Block Execution Example

Status Handling - Bi-directional Edge Trigger Function Block

The output status is set to the input status.

Parameters - Bi-directional Edge Trigger Function Block

The following table lists the system parameters for the Bi-directional Edge Trigger function block:

Bi-directional Edge Trigger Function Block System Parameters

Parameter	Units	Description
IN_D	None	The discrete input value and status.
OUT_D	None	The discrete output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Bi-directional Edge Trigger Function Block

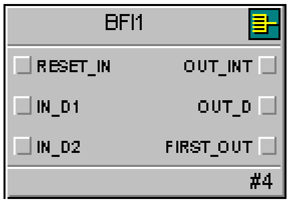
The Bi-directional Edge Trigger function block is used to trigger other logical events based on the transition of a logical signal. For example, you can use the block to trigger actions based on the startup or shutdown of equipment.

Note The output of this block should not be used in a different module unless you are certain that the scan rates match and will not be changed in the future.

You can lengthen the pulse by using an Off-delay Timer function block, or you can use the pulse output to set a latch.

Boolean Fan Input (BFI) Function Block

The Boolean Fan Input (BFI) function block generates a discrete output based on the weighted binary sum, binary coded decimal (BCD) representation, transition state, or logical OR of one to sixteen discrete inputs. The block supports signal status propagation. There are no modes or alarm detection in the Boolean Fan Input function block.



Boolean Fan Input (BFI) Function Block

RESET_IN is the input that, when True (1), clears FIRST_OUT and activates the trap condition after all the inputs go False.

IN_D1 through IN_D[n] are the discrete input values and statuses (as many as 16 inputs).

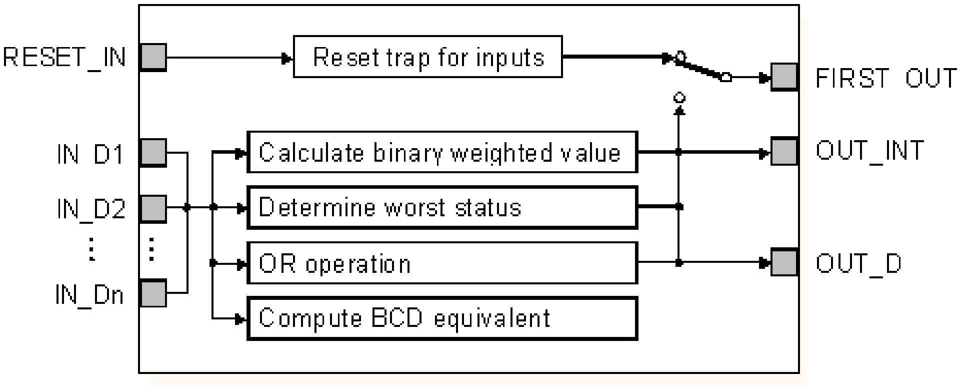
OUT_INT is the unsigned 32-bit binary weighted output value that represents the bit combination of the inputs (IN_D).

OUT_D is the output value that represents the logical OR of the inputs (IN_D).

FIRST_OUT is the binary weighted output of the discrete input values when one or more inputs is set after RESET_IN is set.

Schematic Diagram - Boolean Fan Input Function Block

The following figure shows the internal components of the Boolean Fan Input function block:



Boolean Fan Input Function Block Schematic Diagram

Block Execution - Boolean Fan Input Function Block

The number of inputs to the Boolean Fan Input function block is an extensible parameter. The block default is two inputs. You add inputs by selecting the function block diagram, clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs. This creates additional input connectors for the block.

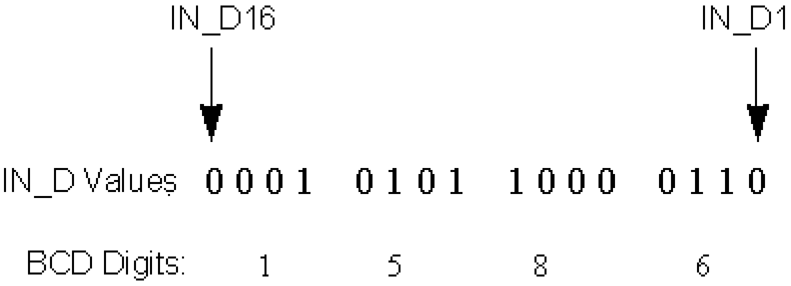
The Boolean Fan Input function block examines the discrete input values for the Set state at each block execution. The OUT_INT output is set to the binary weighted value of the discrete inputs (IN_D1 is weighted as 1, IN_D2 as 2, IN_D3 as 4, IN_D4 as 8). The status of OUT_INT is set to the worst status among the inputs.

When OUT_INT transitions from zero to non-zero and ARM_TRAP is non-zero, a trap condition is flagged in FIRST_OUT by copying the value of OUT_INT to FIRST_OUT. The FIRST_OUT output updates whenever the IN_Dx parameter transitions from all zero to one or more non-zero while ARM_TRAP is non-zero. FIRST_OUT also updates when RESET_IN is non-zero as the block changes FIRST_OUT and RESET_IN back to zero. The status of FIRST_OUT is equal to the status of OUT_INT when the trap occurred.

The value of the OUT_D output is the logical OR of the discrete inputs. Its status is equal to the worst status among the inputs.

To support thumbwheel switch interfaces, the Boolean Fan Input function block uses a contained parameter to store the binary coded decimal (BCD) representation of the discrete inputs. The first four discrete inputs are used to construct the BCD ones digit. (Within this nibble, the first input is the least-significant bit.) The next four inputs are used for the BCD tens, hundreds, thousands, and ten thousands digits. When the four bits representing a digit are greater than nine, the digit is limited to nine.

The following figure is an example of Boolean Fan Input function block execution for OUT_INT = 5510. The result is BCD = 1586 and OUT_D = True.



Boolean Fan Input Function Block Execution Example

Status Handling - Boolean Fan Input Function Block

The OUT_INT and OUT_D statuses are set to the worst status among the inputs. The FIRST_OUT status is the worst status among the inputs when the FIRST_OUT value is written. The FIRST_OUT status is reset when the FIRST_OUT value is cleared.

Parameters - Boolean Fan Input Function Block

The following table lists the system parameters for the Boolean Fan Input function block:

Boolean Fan Input Function Block System Parameters

Parameter	Units	Description
ARM_TRAP	None	When True (1), ARM_TRAP enables the first out trap mechanism.
BCD	None	The binary coded decimal representation of the discrete inputs.
FIRST_OUT	None	The binary weighted output when one or more inputs is set after RESET_IN is set.
IN_D1 to IN_D16	None	The discrete input values and statuses. The number of inputs is an extensible parameter.
OUT_D	None	The output value that represents the logical OR of the discrete inputs.
OUT_INT	None	The unsigned 32-bit binary weighted output value that represents the bit combination of the inputs.
RESET_IN	None	When True (1), RESET_IN clears FIRST_OUT. The trap condition is not activated again until all inputs go False. The block sets RESET_IN back to zero at the end of each scan.

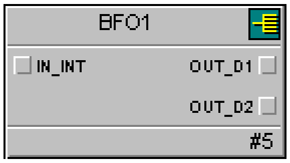
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Boolean Fan Input Function Block

The Boolean Fan Input function block is useful to detect and trap one or more of the discrete inputs as it transitions to the Set state. When this condition is detected, a copy of the OUT_INT output is held in the block's FIRST_OUT output. This output can be used to determine which input(s) or initial conditions caused a machine to fail. The OUT_INT output always reflects the state of the inputs, whereas the FIRST_OUT output updates whenever the IN_Dx parameter transitions from **all zero** to **one or more non-zero**.

Boolean Fan Output (BFO) Function Block

The Boolean Fan Output (BFO) function block decodes a binary weighted input to individual bits and generates a discrete output value for each bit. The block supports signal status propagation. There are no modes or alarm detection in the Boolean Fan Output function block.



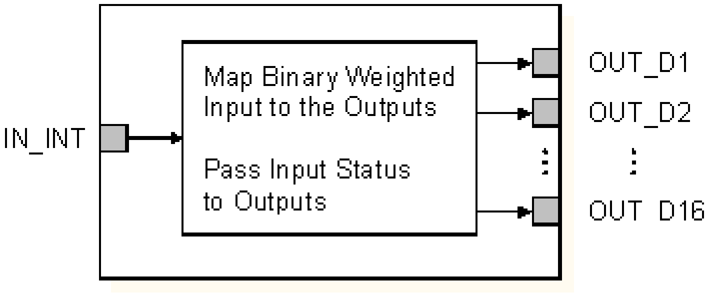
Boolean Fan Output (BFO) Function Block

IN_INT is the unsigned 32-bit binary weighted input value and status.

OUT_D1 through OUT_D[n] are the discrete output values and statuses (as many as 16 outputs) that represent the bit of the input.

Schematic Diagram - Boolean Fan Output Function Block

The following figure shows the internal components of the Boolean Fan Output function block:


IN_INT
Map Binary Weighted Input to the Outputs
Pass Input Status to Outputs
OUT_D1
OUT_D2
OUT_D16

Boolean Fan Output Function Block Schematic Diagram

Block Execution - Boolean Fan Output Function Block

The Boolean Fan Output function block treats the unsigned 32-bit input as a binary weighted value. The individual bits that comprise this value are translated to the block's discrete outputs.

The number of outputs in the Boolean Fan Output function block is an extensible parameter. The block default is two outputs. You add outputs by selecting the function block diagram, clicking the right mouse button, and clicking Extensible Parameters, and modifying the number of outputs. This creates additional output connectors for the block.

The first discrete output represents the least-significant bit of the translated input value. The second discrete output is the next least-significant bit, and so on. The status of the input (IN_INT) is passed to the statuses of the discrete outputs (OUT_D).

The following is an example of Boolean Fan Output function block execution for IN_INT = 5153.

0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1
OUT _D16	OUT _D15	OUT _D14	OUT _D13	OUT _D12	OUT _D11	OUT _D10	OUT _D9	OUT _D8	OUT _D7	OUT_ D6	OUT_ D5	OUT_ D4	OUT_ D3	OUT_ D2	OUT_ D1

Status Handling - Boolean Fan Output Function Block

The statuses of the block outputs (OUT_D) are set equal to the status of the block input (IN_INT).

Parameters - Boolean Fan Output Function Block

The following table lists the system parameters for the Boolean Fan Output function block:

Boolean Fan Output Function Block System Parameters

Parameter	Units	Description
IN_INT	None	The unsigned 32-bit binary weighted input value and status.
OUT_D1 to OUT_D16	None	The discrete output values and statuses. The number of outputs is an extensible parameter.

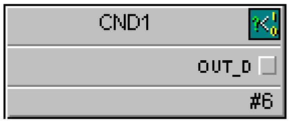
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information

You can use the Boolean Fan Output function block to optimize data communication between two controllers. To do this, you use a Boolean Fan Input block on one controller to condense discrete values into an integer. Next, you send the integer value to a Boolean Fan Output function block on the other controller to expand the integer to its original discrete representation.

Condition Function Block

The Condition (CND) function block evaluates a single-line expression and generates a discrete output value when the expression is evaluated True (1) for longer than a specified time period. Mathematical functions, logical operators, and constants can be used in the expression. There are no modes or alarm detection in the Condition function block.

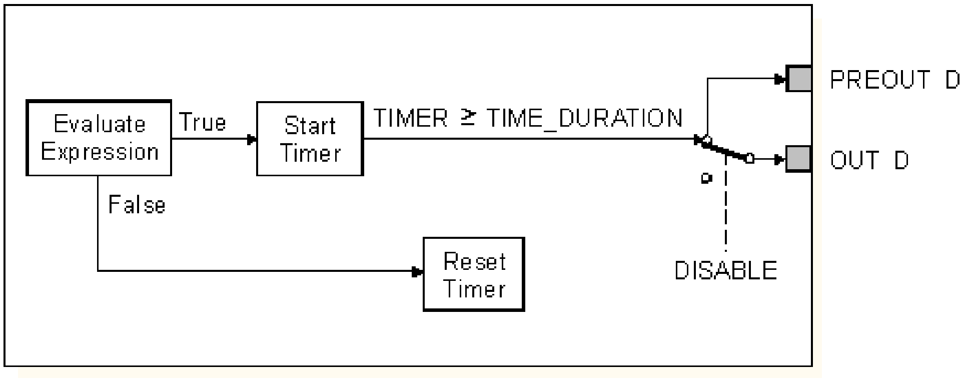


Condition Function Block

OUT_D is the discrete output value and status.

Schematic Diagram - Condition Function Block

The following figure shows the internal components of the Condition function block:



Condition Function Block Schematic Diagram

Block Execution - Condition Function Block

Expressions are structured text that is in a specific syntax. You write expressions using the Expression Editor. An expression is made up of operands, operators, functions, constants, and keywords.

When the expression in the Condition function block is evaluated True (1) for longer than the configured TIME_DURATION, the OUT_D output and the PRE_OUT_D parameter are set. When the expression is True but changes to False (0) before the TIME_DURATION is reached, the timer is reset until the expression evaluates True again. An expression that evaluates to zero resets OUT_D and PRE_OUT_D.

The DISABLE parameter is used to bypass the block logic. When DISABLE is True, the OUT_D parameter is not set, regardless of the expression evaluation. However, the PRE_OUT_D parameter is set and can be used elsewhere.

Refer to the Expressions topic for syntax guidelines for writing expressions and for information on the supported operands, operators, functions, constants, and keywords.

Status Handling - Condition Function Block

The PRE_OUT_D and OUT_D statuses are normally GoodNonCascade. If any part of the expression is not resolved in run time, the Condition function block sets the PRE_OUT_D and OUT_D statuses to Bad: Configuration Error. The whole expression is never evaluated as True if part of the expression is not resolved. If the expression is resolved a read error is encountered (for example, when a parameter is referenced in a non-communicating controller), the status of PRE_OUT_D and OUT_D depends on ALGO_OPTS. If the Abort on Read Errors option is selected, the status of PRE_OUT_D and OUT_D remains GoodNonCascade. Otherwise, the status of PRE_OUT_D and OUT_D is BadNoComm.

Parameters - Condition Function Block

The following table lists the system parameters for the Condition function block:

Condition Function Block System Parameters

Parameter	Units	Description
ALGO_OPTS	None	Algorithm options. When selected, the expression algorithm will abort after a read error. The Algorithm option is AbortOnReadErrors. When the Condition block expression aborts, the value and status of PRE_OUT_D and OUT_D remain unchanged.
ERROR_OPT	None	Specifies how the block behaves when a read error occurs. The value of PRE_OUT_D will be False (default), True, or the last value prior to the read error as defined in ERROR_OPT (unless ‘Abort on Read Errors’ is set in ALGO_OPTS, in which case ERROR_OPT does not apply). The status of PRE_OUT_D is BadNoComm when a read error occurs.
DESC	None	User-specified description of the expression.
DISABLE	None	Enables/disables the block logic (True = disable, False = enable).
OUT_D	None	The discrete output value and status.
PRE_OUT_D	None	The internal value set when the expression evaluates to True for the period defined by TIME_DURATION. PRE_OUT_D ignores the DISABLE value.
TIME_DURATION	Seconds	Specifies the time the expression must be True to set OUT_D.
TIMER	Seconds	The time the expression is True.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

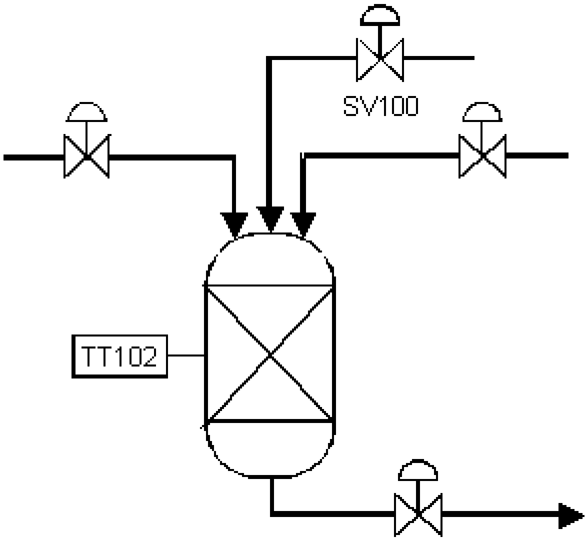
Application Information - Condition Function Block

You can use the Condition function block to implement complex calculations, signal conditioning, or functionality that is not available in the standard function blocks. You can use trigonometric, exponential, and power functions, as well as standard mathematical operations.

You can also use a group of Condition function blocks to evaluate multiple conditions and to send the outputs to other logical function blocks, such as the And, Or, or Boolean Fan Input (BFI) function blocks.

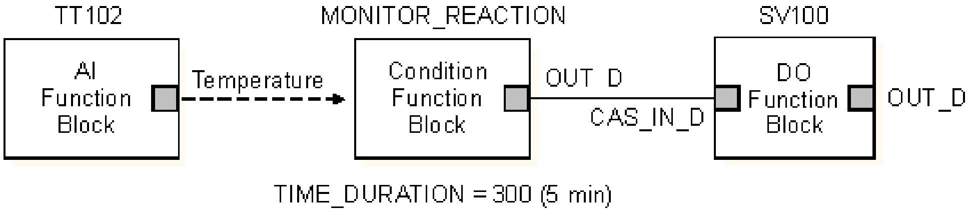
Application Example: Monitor Exothermic Runaway

Assume you have a chemical reactor that combines ingredients for an exothermic reaction. The process must be monitored for the presence of a runaway exothermic reaction. When one occurs, the flushing system must start up automatically to bring the reaction under control. The following figure shows an example of this process:



Condition Function Block Application Example

Assume the reaction is considered a runaway when the temperature is greater than 400°F for longer than five minutes. When this condition occurs, the SV100 valve must be opened to allow operation of the flushing system. You use a Condition function block to monitor for this condition and to send a signal to start the flushing operation. The following figure is the function block diagram for this example:



Condition Function Block Diagram Example

When the temperature exceeds 400°F, the condition evaluates as True (1). When the condition remains True for more than 5 minutes (300 sec), OUT_D is set True. This signals the DO function block to open the SV100 valve and begin the flushing operation. You configure the following expression in the Condition function block for this application:

```
REM Return True when reaction temperature is greater than 400
REM degrees, otherwise return False
'/TT102/PV.CV' > 400.0
```

When the reaction is slowed and the temperature drops below 400°F, the condition evaluates as False (0) and OUT_D is False. This signals the DO function block to shut the flushing system valve.

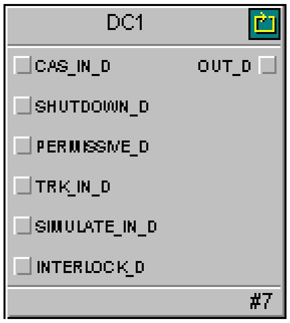
Device Control Function Block

The Device Control (DC) function block provides setpoint control for multistate discrete devices, such as motors, pumps, and block valves. The block compares the requested state (setpoint) to the actual state reported from the device and, after allowing time for the device to change state, detects alarm limits on any error. The basic functionality is augmented by an assortment of interlocks and device control options to customize the block's operation for your application.

The Device Control function block supports mode control, setpoint tracking, simulation, and alarm limit detection. You can select options that specify the control strategy used in the block.

The setpoint requests the device to go to one of two or three supported states: Passive, Active 1 and Active 2 (optional). The Passive state is the power failure (safe) state, such as OFF or CLOSED. An Active state usually requires energy (or allows energy to flow), such as OPEN, RUN, FORWARD, or REVERSE. You configure one or two Active states (Active 1 and Active 2) to match the device you want to control. You select the set of state names that applies to the device, such as STOP/FORWARD/REVERSE or OFF/LOW/HIGH.

The Device Control block uses as many as eight discrete I/O channels to command a device to the requested setpoint state and to read back its confirmation contacts. Discrete I/O is associated with the Passive and Active states by means of a mask for each state that allows each bit to be defined as True (1), False (0), or not used. You can configure four bits as outputs to the device and four bits as the contacts that confirm the device state. The confirm contacts must be maintained because the block is designed to alarm on loss of confirmation.



Device Control Function Block

CAS_IN_D is the discrete value and status of a setpoint from another block used when the block is in Cascade mode.

SHUTDOWN_D is the Emergency Stop discrete value and status input that forces and holds the device in the Passive state.

PERMISSIVE_D is the optional discrete input value and status that must be True when the Permissive device option is enabled to allow the device to be commanded to an Active state.

TRK_IN_D is the discrete input value and status that forces the block mode to Local Override and causes the output to follow the field value (FV_D).

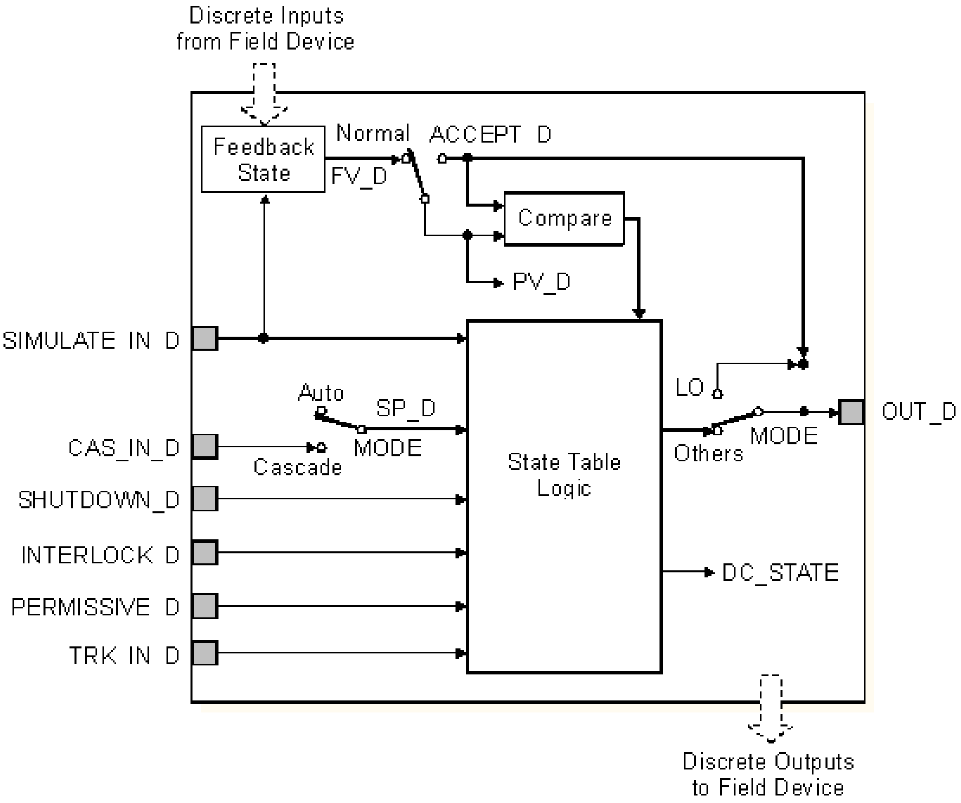
SIMULATE_IN_D is the optional discrete input value and status that is used to simulate the field value.

INTERLOCK_D is the optional discrete input value and status that must be True when the Interlock device option is enabled for the device to remain in the Active state.

OUT_D is the discrete output value and status that reflects the current commanded device state.

Schematic Diagram - Device Control Function Block

The following figure shows a simplified view of the internal components of the Device Control function block:



Device Control Function Block Schematic Diagram

Block Execution - Device Control Function Block

The Device Control function block acquires the setpoint (requested state) and determines whether a transition to a new state is required. Next, the block reads the state of the physical device (feedback state) and compares it to the setpoint. The state of the device controller (DC_STATE) is determined from this comparison, and a failure is generated when the DC_STATE does not match the setpoint. Finally, the output state is calculated and written.

You assign inputs and outputs that correspond to your discrete hardware connections, and you configure state masks that define the expected or desired input/output values for different states. You can assign as many as four feedback signals to monitor field device operation.

You can select setpoint tracking, simulation, and/or control logic options to customize the block for your particular device, such as a pneumatic valve, motorized valve, or other motor.

Determining the Command Setpoint (Requested State)

The block must first calculate its setpoint (SP_D) based on the following logic:

- When the mode is Cascade (Cas), the setpoint is copied from CAS_IN_D.
- When the mode is Automatic (Auto), the setpoint remains where it was left at the last execution of the block or at the last operator entry. Next, if the tracking parameter (TRK_IN_D) is True, the actual mode is changed to Local Override (LO) and the output (OUT_D) is copied from the field confirm value (FV_D) when it is in a valid state (not Undefined). If the SP Track device option is True in this case, the setpoint is copied from the output (OUT_D).
- When the SP Track device option is True and the actual mode is LO due to an interlock or shutdown condition, the setpoint is copied from the output (OUT_D).

You can select the Passive on Active Timeout device option to cause the OUT_D output to be changed to the Passive state during the execution of the block when the Active confirm timer parameter times out.

In addition, you can select the Trip device option. When the Trip device option is True and an active confirm is lost for a time greater than the trip time parameter (TRIP_TIME), OUT_D is set to the Passive state. The SP_D value must be changed to Passive to clear the tripped condition.

Determining if a Transition is Required

The setpoint is used next by the logic that determines the device controller state (DC_STATE) to see whether or not a transition to another state is required. A Passive state setpoint always causes the output to be Passive. An Active state setpoint does not necessarily cause the output to go to the desired state.

You can select the Permissive device option if you want a permissive to be required before the state changes. When the Permissive device option is True, the PERMISSIVE_D input must be True for an Active state setpoint to change the DC_STATE to that Active state. However, PERMISSIVE_D does not have to be True before SP_D can be changed to an Active state.

PERMISSIVE_D has no further effect once DC_STATE is in an Active state. When DC_STATE changes away from the Active state, PERMISSIVE_D must be True to return to that state, even if SP_D has not changed.

Delaying Setpoint Changes

When the value of DELAY_TIME is not zero, a change in setpoint from the Passive state to an Active state is delayed for DELAY_TIME seconds (using the DELAY_TIMER) before it is sent to the logic that determines DC_STATE. This allows a common external setpoint to cause a sequenced start of a group of motors.

When the value of RESTART_TIME is not zero, a change in setpoint from one Active state to the same or another Active state is delayed for the time specified in the RESTART_TIME parameter (using DELAY_TIMER). The value of OUT_D is held at the Passive state while DELAY_TIMER is active. This allows time for a motor to stop before reversing direction, or for a compressor to unload before restarting.

DELAY_TIMER is set to the delay or restart time at the beginning of the interval and decreases to zero. You can see the time remaining for the delay or restart to be complete in the DELAY_TIMER parameter value.

Determining the Physical Device State (Feedback State)

The state of the physical device must be reported back to the device controller so that it can be compared to the setpoint. This feedback state is shown in the FV_D parameter. The use of binary discrete inputs to determine FV_D is described in the Assigning I/O section below.

It is possible for the state to be Undefined when the device is moving between steady states. You can specify the maximum amount of time that the state can be Undefined before declaring the move to be a failure by configuring the following parameters:

- CFM_ACT1_TIME is the maximum time allowed to transition to the Active 1 state.
- CFM_ACT2_TIME is the maximum time allowed to transition to the Active 2 state.
- CFM_PASS_TIME is the maximum time allowed to transition to the Passive state.

Timing takes place in the TRAVEL_TIMER (Travel is the term used to describe the movement of a valve from one end of its stroke to the other.). This timer is reset to zero at the start of the transition and increases in value until the device is confirmed or the confirm time has expired. The travel time is left in the timer until the next transition occurs so you can gather statistics for a preventive maintenance program.

Some valves take a long time to travel. When a valve begins to open, it is said to **crack**. You set CRACK_TIME to the longest time it should take to lose the confirm signal for the previous state. This is much shorter than the travel time, so it gives the operator an early warning that the valve is not moving. This time appears in CRACK_TIMER, which increases in value and holds the time until the next transition.

Generally, a device takes a finite amount of time to arrive at a newly commanded state. The Device Control function block supports and tracks these transitions with the DC_STATE parameter. DC_STATE represents the current state of the device. The following table lists the index values for DC_STATE.

Device Control Function Block DC_STATE Parameter Values

Type of State	DC_STATE Index	Meaning
Steady States	0	Confirmed Passive
	1	Confirmed Active 1
	2	Confirmed Active 2
Transient States	3	Going to Passive
	4	Going to Active 1
	5	Going to Active 2
Failure States	6	Failed Passive
	7	Failed Active 1
	8	Failed Active 2
Special States	9	Tripped
	10	Shutdown/Interlocked
	11	Locked

The PV_D value is used to determine DC_STATE. PV_D is normally a copy of FV_D. However, sometimes the hardware that confirms the device state fails. In this case, the operator can confirm the state of the field device visually. When the field device is in the correct state, the operator can turn on the accept switch (ACCEPT_D = True).

When ACCEPT_D transitions from False to True, PV_D is copied from OUT_D and is held there until OUT_D changes state. You can configure graphic displays and process control logic to use PV_D as the confirm state even when a confirm switch fails. The operator can turn off the visual confirmation by changing SP_D or by manually setting ACCEPT_D to False.

Determining the Device Controller State (DC_STATE)

In general, the effect of a change in setpoint is to change the state of the device controller (shown by DC_STATE) to one of the three normal steady states: Passive, Active 1, or Active 2. However, there are other possible states.

- Three transient states corresponding to the transitions between steady states
- Three failure states corresponding to loss of confirm for a steady state or to failure to complete a transition to that state within the time allowed
- Three special states

You can select the Interlock device option. When the Interlock device option is True, the INTERLOCK_D input must be True for DC_STATE to remain in an Active state. When INTERLOCK_D transitions to False:

- The actual mode is changed to LO
- DC_STATE is changed to Shutdown/Interlocked

  and
- OUT_D is set to the Passive state

When INTERLOCK_D becomes True again, DC_STATE might return to its former state if the SP Track device option is False. There is no delay on the return.

When the SHUTDOWN_D input becomes True, the same thing happens as described above for loss of the interlock input. This is not optional.

When you have motor starters with overload protection that must be reset manually, you might want to ensure that resetting the overload does not start the motor until the operator is ready to start it. When the Trip device option is True and an Active state confirm is lost for more than TRIP_TIME seconds, DC_STATE changes to Tripped and OUT_D is set to Passive. The operator (or block logic in Cas mode) must write the SP_D to Passive to leave the Tripped state before writing the SP_D to the former active state to resume that state.

You can select the Reset Required device option. This device option works with shutdown, interlock, and trip. If the Reset Required device option is True, DC_STATE changes to Locked when you clear a device state of shutdown, interlocked, or tripped.

The following is an example of how this device option works with shutdown.

Condition	DC_STATE	FAIL
SP_D = 1	Confirmed Active 1	Clear
SHUTDOWN_D = 1	Shutdown/Interlocked	Passive Confirm Time
SHUTDOWN_D = 0	Locked	Passive Confirm Time
RESET_D = 1	Confirmed Active 1	Clear

A motor trip goes to the Tripped state, but requires reset and a manually entered Passive setpoint value. The RESET_D parameter must be turned on to clear the Locked state and go to Passive. A Locked state due to shutdown or interlock does not require a manually entered Passive setpoint input to resume the Active state. When the shutdown or interlock condition clears and RESET_D is set True, the previous Active state resumes.

By default, RESET_D has the Control lock assignment and this allows an operator to change it. If your application requires a supervisor to reset the device controller when something abnormal happens, you can change the parameter lock assignment on RESET_D. Refer to the Parameter and Function Security topic for more information on how to change parameter lock assignments.

Determining the Failure Code

When the state of the device controller in DC_STATE does not match the setpoint, a failure code is generated. This code is stored in the FAIL parameter. The FAIL_ACTIVE parameter is set True when the value of FAIL is not zero. The following failures are possible.

Device Control Function Block FAIL Parameter Codes

FAIL Parameter Code	Meaning	Cause
0	Clear	
1	Passive Confirm Time	Set when OUT_D is set Passive and FV_D does not change to Passive in the required time duration or if FV_D does not change state before CRACK_TIMER times out.
2	Active 1 Confirm Time	Set when SP_D sets OUT_D to Active 1 and FV_D does not change to Active 1 in the required time duration or FV_D does not change state before CRACK_TIMER times out.
3	Active 2 Confirm Time	Set when SP_D sets OUT_D to Active 2 and FV_D does not change to Active 2 in the required time duration or FV_D does not change state before CRACK_TIMER times out.
4	Passive Confirm Lost	Set when OUT_D and FV_D are Passive and FV_D changes to another state.
5	Active 1 Confirm Lost	Set when OUT_D and FV_D are Active 1 and FV_D changes to another state.
6	Active 2 Confirm Lost	Set when OUT_D and FV_D are Active 2 and FV_D changes to another state.
7	Tripped	Set when DC_STATE is set to Tripped.
8	Shutdown/Interlock	Set when DC_STATE is set to Shutdown/Interlock.

Determining the Output State

The state of OUT_D is the same as DC_STATE for the steady, transient, and failure states. For example, when the setpoint causes DC_STATE to change to Active 1, DC_STATE becomes Going to Active 1. OUT_D becomes Active 1 and stays there when DC_STATE changes to Confirmed Active 1 or Failed Active 1. This behavior is required when the device is a pneumatic valve.

When the Trip or Reset Required device option is True, the failed state is not entered and OUT_D returns to the Passive state. This behavior is required when the device is a motor.

The three special states all set OUT_D to Passive.

When the Passive when Confirmed device option is True, the state of OUT_D is Passive in either of the Active states of DC_STATE or in the failed states. This allows a motorized valve to be energized only during the transition time; the motor must not run when travel is complete.

Assigning I/O

The Device Control function block allows as many as four feedback signals and four output signals to be associated with the discrete field device. The eight input/output signals are referenced by the Device Control block through the I/O parameters. The IOREF_IN[1-4] parameters reference the discrete feedback signals coming from the field device. The IO_OUT[1-4] parameters reference the discrete output signals that command the field device to a requested state.

The feedback input IOREF_IN[1-4] discrete signal values and statuses are stored in the internal parameters F_IN_D[1-4]. These signal values are used for FV_D state determination.

Note The F_IN_D parameters can be exposed and wired to instead of the corresponding IO_IN parameters to connect module-level parameters into the PV of a DC block. For a PV that is derived from a combination of DST signals and module parameters, we recommend using DI blocks to read the DST signals into the module and wire all signals to the F_IN_D parameters of the DC Block. If IO_IN is used in conjunction with F_IN_D parameters, the user must configure the DST signals starting at IO_IN1, IO_IN2, and so forth, and use the remaining inputs for the F_IN_D parameters.

Note A DC block with the Output Failure value selected in the BAD_MASK parameter, only sets the BAD_ACTIVE flag on output failure if the associated I/O channel was at one time enabled, then disabled, and then the I/O card *only* is downloaded.

The requested state (stored in OUT_D) is decoded to the individual IO_OUT[1-4] channel outputs according to the corresponding output state mask. The value and status of the IOREF_OUT[1-4] channels are read back and copied to the internal parameters F_OUT_D[1-4] to track the output channel values and to calculate the status of OUT_D.

The IO_OUT[1-4] discrete output channel values are determined from the output mask that corresponds to the current state stored in OUT_D. Likewise, the IO_IN[1-4] discrete input channel values are used to find a matching input mask to determine the state to be stored in FV_D and PV_D.

The STATE_MASKS parameter is used to define the output and input combinations of IO_OUT[1-4] and IO_IN[1-4] for each of the three different states that can be contained in the parameters SP_D, OUT_D, FV_D and PV_D.

The SP_D and OUT_D parameters can be one of the three following states.

Value	State
0	Passive
1	Active 1
2	Active 2

The FV_D and PV_D parameters can be one of the four following states.

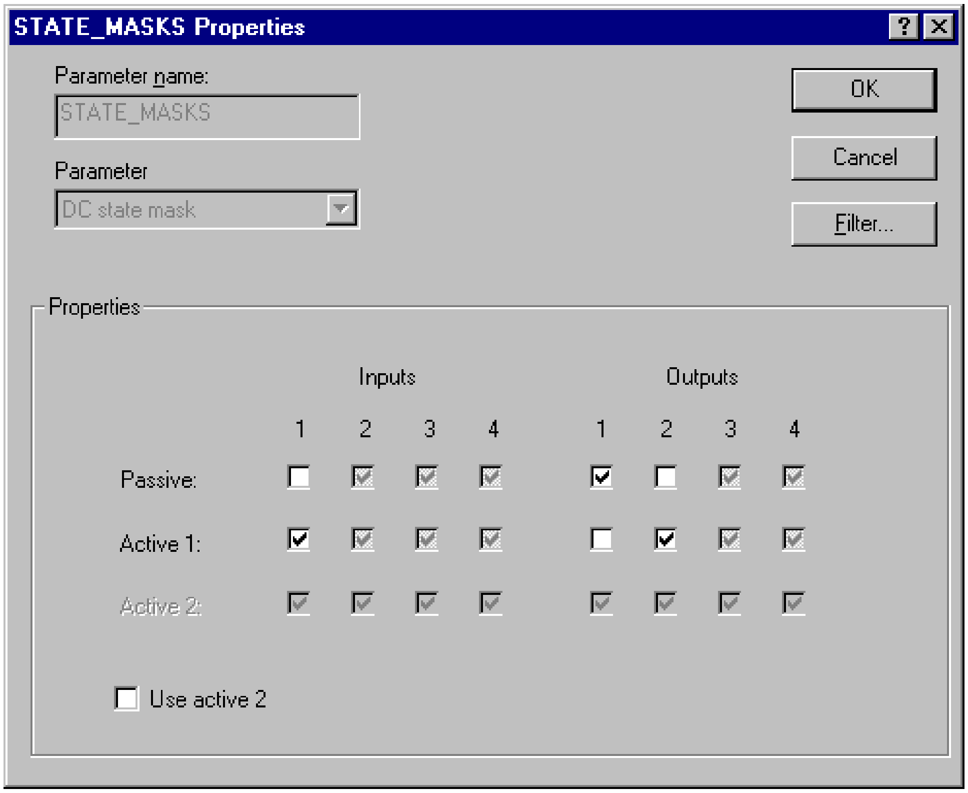
Value	State
255	Undefined
0	Passive
1	Active 1
2	Active 2

The Undefined state means the IO_IN[1-4] references do not match any of the defined input STATE_MASKS.

Configuring State Masks

You enter the expected/desired values of inputs and outputs for each of three states in the boxes in the State Masks dialog. A blank box indicates a False (0) input or output value. A box with a check mark indicates a True (1) input or output value. Gray boxes indicate inputs and outputs that are not used or do not matter.

The following figure shows a simple state mask example.



STATE_MASKS Properties

Parameter name: STATE_MASKS

Parameter: DC state mask

Properties

	Inputs				Outputs			
	1	2	3	4	1	2	3	4
Passive:	□	▨	▨	▨	☑	□	▨	▨
Active 1:	☑	▨	▨	▨	□	☑	▨	▨
Active 2:	▨	▨	▨	▨	▨	▨	▨	▨

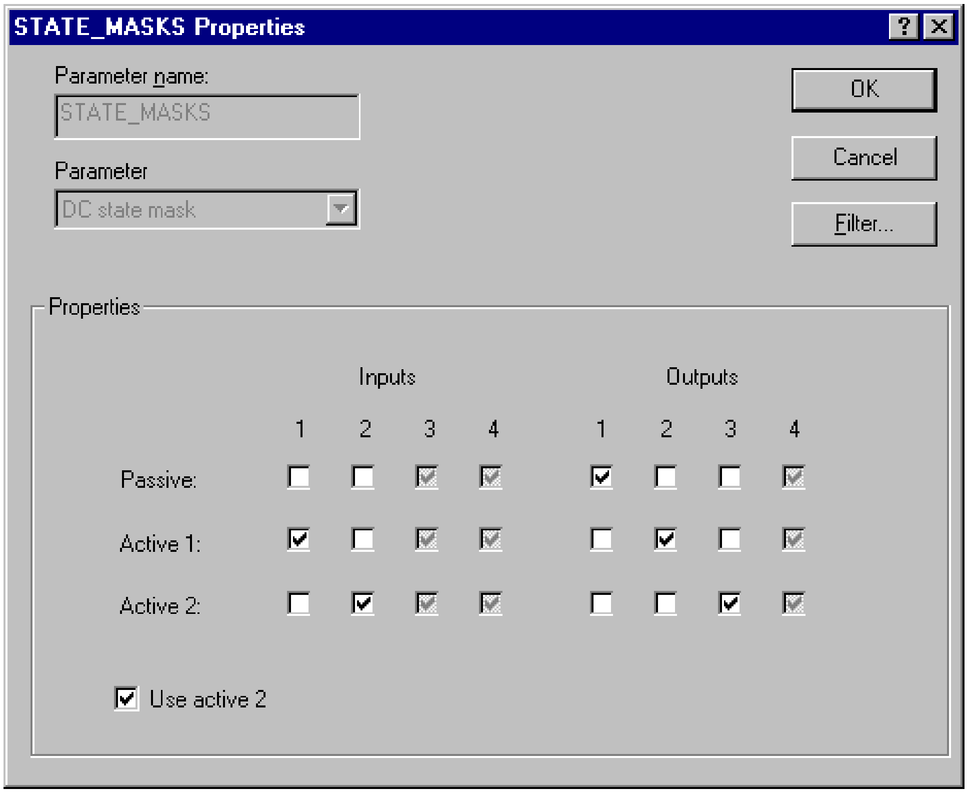
□ Use active 2

Device Control Function Block State Mask Example (No Active 2 State)

In this example, if IO_IN_1 is True (1), the FV_D parameter value is 1, indicating the Active 1 state. When the block sets OUT_D to a value of 1 to request the Active 1 state, the Active 1 output mask is used to set IO_OUT_1 to 0 and IO_OUT_2 to 1.

You mark the Use active 2 box to indicate the Active 2 state is to be used. This is used only to determine whether the Active 2 state is considered when matching bit patterns.

The following figure shows an example definition of the Passive, Active 1, and Active 2 bit masks for a group of input and output I/O parameters.



Parameter name: STATE_MASKS

Parameter: DC state mask

	Inputs				Outputs			
	1	2	3	4	1	2	3	4
Passive:	□	□	☑	☑	☑	□	□	☑
Active 1:	☑	□	☑	☑	□	☑	□	☑
Active 2:	□	☑	☑	☑	□	□	☑	☑

☑ Use active 2

Device Control Function Block State Mask Example (with Active 2 State)

In this example, if IO_IN_1 is False and IO_IN_2 is True, the FV_D parameter value is 2, indicating the Active 2 state. When the block sets OUT_D to a value of 2 to request the Active 2 state, the outputs are set as follows:

IO_OUT_1 = False (0)

IO_OUT_2 = False (0)

IO_OUT_3 = True (1)

IO_OUT_4 = Grayed out (0)

The grayed out output settings are always set to False.

Simulation

To support testing, you can enable simulation. This allows the measurement value and status to be supplied manually or from another block.

During configuration, decide whether you want the simulated value/status to be entered manually during operation or you will use a value/status from another block for the simulated value/status.

When the value is entered manually:

- The operator first enables simulation by selecting the SIMULATE_D parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.
- If SIMULATE_IN_D is not connected (status = Bad: NotConnected), the operator enters the value to be used in the SIMULATE_D parameter Simulate Value field. In online operation, the operator can enter a simulated status value in the Simulate Status field.

Note Make sure SIMULATE_IN_D is not connected if you want to enter the value or status manually. When SIMULATE_IN_D is connected, the value from the SIMULATE_IN_D Value field is used as the simulated value.

When the value/status from another block is used:

- During configuration, connect SIMULATE_IN_D to the desired block output or parameter. Do not enter a value in the Simulate Value field of the SIMULATE_IN_D input; the block uses the connected value automatically.
- During operation, the operator enables simulation by selecting the SIMULATE_D parameter and setting the Simulate Enabled box in the Simulate Enabled/Disabled field.

Note Do not enter a value for the SIMULATE_IN_D parameter. When you do and the status of SIMULATE_IN_D is not Bad: NotConnected, the manually entered value for SIMULATE_IN_D overrides any value you enter in SIMULATE_D.

Tracking

The TRK_IN_D input is used to force the OUT_D to track FV_D. When TRK_IN_D is True, OUT_D is set equal to FV_D when it is in Passive, Active 1, or Active 2 state. The actual mode of the block is Local Override (LO). If the SP track device option is True, SP_D is copied from OUT_D.

The value of PV_D usually is set to the value of FV_D. However, when the ACCEPT_D parameter is True, PV_D is set to the value of OUT_D. Whenever OUT_D changes from its previous value, the block automatically sets ACCEPT_D to False.

Device Options

The device options (DEVICE_OPTS) parameter allows you to select the optional control logic used in the block. You select one or more of the device options by selecting that option in the DEVICE_OPTS Parameter Properties Dialog Box. The following device options are available:

- Passive on Active Timeout
- Passive when Confirmed
- Trip
- Reset Required
- Permissive
- SP Track
- Interlock

With the Passive on Active Timeout device option, a confirm time-out alarm is generated and the DC_STATE is changed to the failed Active state. In Auto mode, SP_D must be manually entered to attempt the Active state again. In Cas mode, SP_D must be changed to Passive and then back to the Active state to attempt the Active state again.

The Passive when Confirmed device option is used for devices, such as a motor operated valve, that require the drive signal to the device to stop when the requested state is reached.

With the Permissive device option, an OUT_D change from Active to Passive is never blocked.

Block Errors

The following conditions are reported in the BLOCK_ERR parameter:

Simulate active – Simulation is enabled and the block is using a simulated value in its execution.

Input failure/process variable has Bad status – The hardware is bad.

Output failure – The output is not valid.

Readback failed – The I/O readback failed.

Modes - Device Control Function Block

The Device Control function block supports two modes:

Automatic (Auto)

Cascade (Cas)

Mode transitions are standard. The mode of the device is specified through the MODE parameter field.

The actual mode changes to Local Override (LO) in the following instances:

- when TRK_IN_D is True
- when SHUTDOWN_D is True
- when the interlock is lost (the Interlock device option is enabled and INTERLOCK_D is False)
- when there is no permit (the Permissive device option is enabled and PERMISSIVE_D is False) and the device state is Confirmed Passive, Going to Passive, or Failed Passive

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Alarm Detection - Device Control Function Block

You can configure the alarm limits for the following standard alarms:

CFM_ACT1_TIME – The maximum time allowed to change from any state to the Active 1 state.

CFM_ACT2_TIME – The maximum time allowed to change from any state to the Active 2 state.

CFM_PASS_TIME – The maximum time allowed to change from any state to the Passive state.

CRACK_TIME – The maximum time allowed to lose the confirm for the state previous to this transition.

DELAY_TIME – The time to delay a change in setpoint to an Active state from any other state. There is no delay going to the Passive state.

RESTART_TIME – The time for OUT_D to be Passive between successive Active states.

TRIP_TIME – The time that an Active confirm can be lost before the confirm is considered to be lost.

Parameters - Device Control Function Block

The following table lists the system parameters for the Device Control function block:

Device Control Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
ACCEPT_D	None	Causes PV_D to become a copy of OUT_D until OUT_D is changed. ACCEPT_D is used to force the block logic to accept the requested state as confirmed without using an actual feedback. The parameter resets when OUT_D changes. **Note** Set this parameter after requesting a new state to simulate the feedback changing to the correct state
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the Device Control function block are Simulate active, Input failure/process variable has Bad status, Output failure, and Readback failed.
CAS_IN_D	None	The remote discrete setpoint value from another block.
CFM_ACT1_TIME	Seconds	The maximum time allowed to transition to the Active 1 state.
CFM_ACT2_TIME	Seconds	The maximum time allowed to transition to the Active 2 state.
CFM_PASS_TIME	Seconds	The maximum time allowed to transition to the Passive state.
CRACK_TIME	Seconds	The maximum time allowed to drop the confirm PV_D of the previous state.

Parameter	Units	Description
CRACK_TIMER	Seconds	The elapsed time of the CRACK_TIME setting. When the device *cracks* or changes state through the FV_D parameter, CRACK_TIMER is held at that value. CRACK_TIMER is reset to zero when SP_D is changed.
DC_STATE	None	The current state of the device controller.
DELAY_TIME	Seconds	Specifies the delay time when SP_D is changed from the Passive state to an Active state before the change is sent to DC_STATE logic. This allows a common external setpoint to cause a sequenced start of a group of motors. A change in SP_D to the Passive state or from one Active state to another is not delayed.
DELAY_TIMER	Seconds	Elapsed time of DELAY_TIME or RESTART_TIME setting. DELAY_TIMER counts down from DELAY_TIME or RESTART_TIME to zero.
DEVICE_OPTS	None	Device options. Allow you to select control logic options used in the block. The device options are Interlock, Passive on Active Timeout, Passive when Confirmed, Permissive, Reset Required, SP Track, and Trip.
F_IN_D1	None	Current value and status of IO_IN_1.
F_IN_D2	None	Current value and status of IO_IN_2.
F_IN_D3	None	Current value and status of IO_IN_3.
F_IN_D4	None	Current value and status of IO_IN_4.
F_OUT_D1	None	Current value and status of IO_OUT_1.
F_OUT_D2	None	Current value and status of IO_OUT_2.
F_OUT_D3	None	Current value and status of IO_OUT_3.
F_OUT_D4	None	Current value and status of IO_OUT_4.
FAIL	None	Named set that indicates the state of the device controller failure detector. Refer to this function block's Block Execution topic for information on failure codes.
FAIL_ACTIVE	None	Binary indication of an active failure found by the failure detector.
FV_D	None	The feedback state of the discrete device. FV_D represents the state of the field device as determined from state masks for the four discrete inputs when SIMULATE_D is disabled (False).
INTERLOCK_D	None	Discrete input that must be True for OUT_D to remain in an Active state when the Interlock device option is enabled.

Parameter	Units	Description
INTERLOCK_OPT	None	Interlock option. Allows you to select the block's behavior when the status of INTERLOCK_D, PERMISSIVE_D, or SHUTDOWN_D is Bad. The three options are:
Always Use Value – The block behaves as described in Block Execution. This is the default value for INTERLOCK_OPT.	Use Last Good Value – The block uses the value of each parameter the last time the parameter's status was not Bad.	Passive if Bad – If the status of any of these parameters is Bad, the block uses the passive state value for that parameter, that is, 0 for INTERLOCK_D and PERMISSIVE_D, and 1 for SHUTDOWN_D
IO_IN_1	None	Input hardware reference that defines the Device Signal Tag (DST) for the first I/O channel used for the feedback field measurement.
IO_IN_2	None	Input hardware reference that defines the DST for the second I/O channel used for the feedback field measurement.
IO_IN_3	None	Input hardware reference that defines the DST for the third I/O channel used for the feedback field measurement.
IO_IN_4	None	Input hardware reference that defines the DST for the fourth I/O channel used for the feedback field measurement.
IO_OUT_1	None	Output hardware reference that defines the DST for the first output I/O channel of the block.
IO_OUT_2	None	Output hardware reference that defines the DST for the second output I/O channel of the block.
IO_OUT_3	None	Output hardware reference that defines the DST for the third output I/O channel of the block.
IO_OUT_4	None	Output hardware reference that defines the DST for the fourth output I/O channel of the block.
MODE	None	The mode record of the block. Contains the actual, target, permitted, and normal modes.
OUT_D	None	The discrete output value and status.
PERMISSIVE_D	None	Discrete input that must be True for OUT_D to transition to an Active state when the Permissive device option is set True.
PV_D	None	The discrete process variable used in block execution. It is a copy of FV_D or SP_D.
RESET_D	None	Unlocks the device controller after a failure when the Reset Required option is enabled. RESET_D unlocks OUT_D from the locked Passive state when Reset Required is enabled.

Parameter	Units	Description
RESTART_TIME	Seconds	Specifies the delay time when SP_D is changed from one Active state to the same or another Active state. The value of OUT_D is held at the Passive state while DELAY_TIMER is active. This allows time for a motor to stop before reversing direction, or for a compressor to unload before restarting.
SHUTDOWN_D	None	Sets and holds OUT_D in the Passive state when SHUTDOWN_D is True.
SIMULATE_D	None	Enables simulation and allows you to enter an FV_D value and status when SIMULATE_IN_D is not connected.
SIMULATE_IN_D	None	The input connector value and status used for FV_D when simulation is enabled.
SP_D	None	The setpoint for the device controller. In Auto mode, SP_D represents the manually entered requested device state. In Cas mode, SP_D comes from the CAS_IN_D input.
STATE_MASKS	None	The command and feedback state definition masks. STATE_MASKS maps combinations of the feedback input signals to the three discrete states and maps each of the three discrete states to combinations of output signals.
TRAVEL_TIMER	Seconds	Counts the elapsed time of travel (the time it took the new requested state to be confirmed). TRAVEL_TIMER is held at the current value when FV_D is changed to the requested state. The timer is reset when the state changes from Confirmed. **Note** TRAVEL_TIMER can be used as a tool to set maximum confirm times.
TRIP_TIME	Seconds	The time allowed before a loss of confirm is interpreted as a trip. For example, when a state was confirmed and the feedback changes state without requesting a new state, the time in TRIP_TIME must pass before the device is labeled Tripped.
TRK_IN_D	None	Discrete input that causes OUT_D to track FV_D.

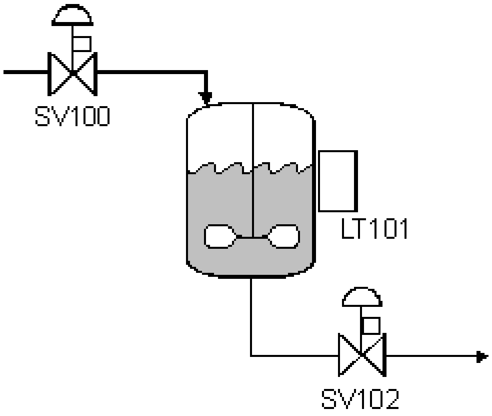
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Device Control Function Block

You can use the Device Control function block for simple or complex discrete control. The following examples show some of the capabilities of the block.

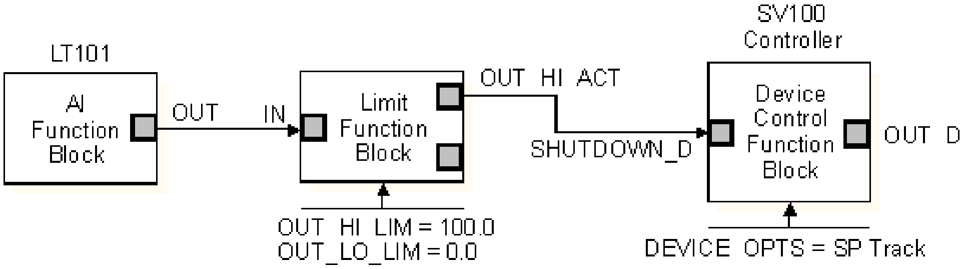
Application Example: Simple Shutdown

Assume you have a solenoid valve that is used to fill a tank. The valve has two positions: OPEN and CLOSED. The following figure illustrates this application:



Device Control Function Block Tank Fill Application Example

Assume you want the operator to be able to open or close the valve when the tank level is less than 100%, but the valve must close when the tank level reaches 100%. When the level reaches 100%, a Limit function block writes a shutdown signal to the Device Control block's SHUTDOWN_D input. SHUTDOWN_D holds the valve in the CLOSED (Passive) state. The following figure is the function block diagram for this example:

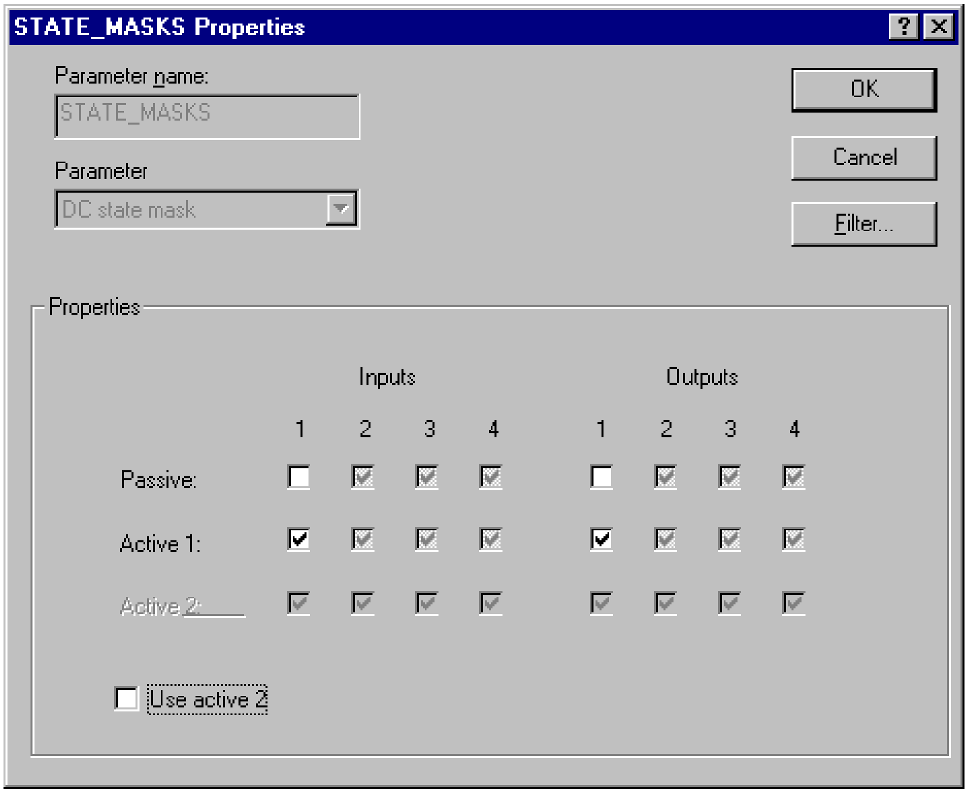


Device Control Function Block Diagram Example for Tank Fill Application

Assume the solenoid valve responds to a single discrete input for its commanded state. A False (0) value represents the close action and a True (1) value represents the open action. You select the SP Track device option to force the SP_D to the CLOSED (Passive) state when the tank is full.

Warning Disabling this option can potentially start the equipment when unattended if the SP is left Active following a failed start due to an interlock.

A single output is used to feedback the actual state of the valve. A False value indicates the valve is CLOSED and a True value indicates the valve is OPEN. You configure the STATE_MASKS parameter to match these signals, as in the following example. The Active 2 state is not used.



STATE_MASKS Properties

Parameter name: STATE_MASKS

Parameter: DC state mask

Properties

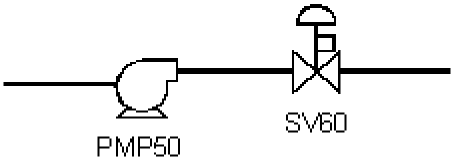
	Inputs				Outputs			
	1	2	3	4	1	2	3	4
Passive:	☐	☑	☑	☑	☐	☑	☑	☑
Active 1:	☑	☑	☑	☑	☑	☑	☑	☑
Active 2:	☑	☑	☑	☑	☑	☑	☑	☑

☐ Use active 2

Device Control Function Block State Mask Example for Tank Fill Application

Application Example: Simple Interlock

Assume you have a pump that you want the operator to turn on only when the downstream valve is open. The following figure illustrates this application:

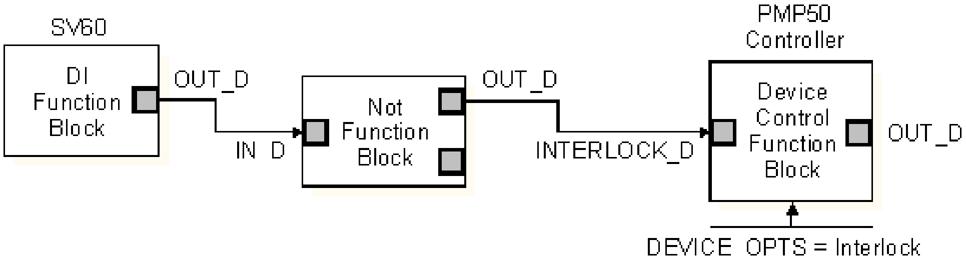


Device Control Function Block Pump Interlock Application Example

The OPEN/CLOSED signal from the valve is used as an input to the Device Control function block's INTERLOCK_D parameter. This prevents the pump from being turned on when the valve is closed.

Warning Disabling this option can potentially start the equipment when unattended if the SP is left Active following a failed start due to an interlock.

The following figure is the function block diagram for this example:



Device Control Function Block Diagram Example for Pump Interlock Application

You select the Interlock device option so the Device Control block considers the INTERLOCK_D parameter in its logic. When the SV60 valve is CLOSED (False [0]), the interlock is Active and the operator request to turn on the pump is held until the SV60 valve is OPEN (True [1]).

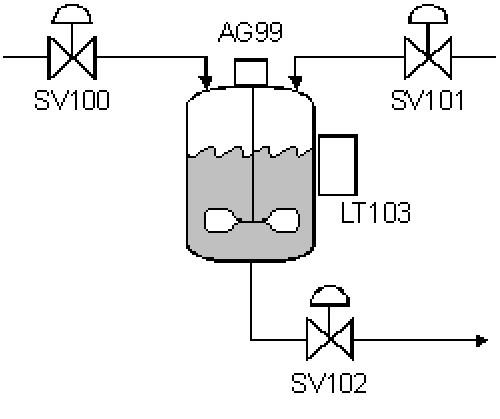
Application Example: Interlock with Permissive

In the following example, the Device Control block is used to control a mixing agitator. The agitator is a three state motor: OFF, SLOW, and FAST. The agitator accepts two discrete input signals for state commands and provides two discrete contracts for state feedback indication.

To prevent foaming of the components being mixed, the operator is not permitted to start agitation until the tank level is 50% or greater. To prevent molecular breakdown, the mixing operation must not exceed 30 minutes when mixed at the slow speed or 15 minutes at the fast speed. After the components are mixed, the operator must be locked out from reactivating the agitator until the mixture is drained and the tank is refilled with the next batch of components.

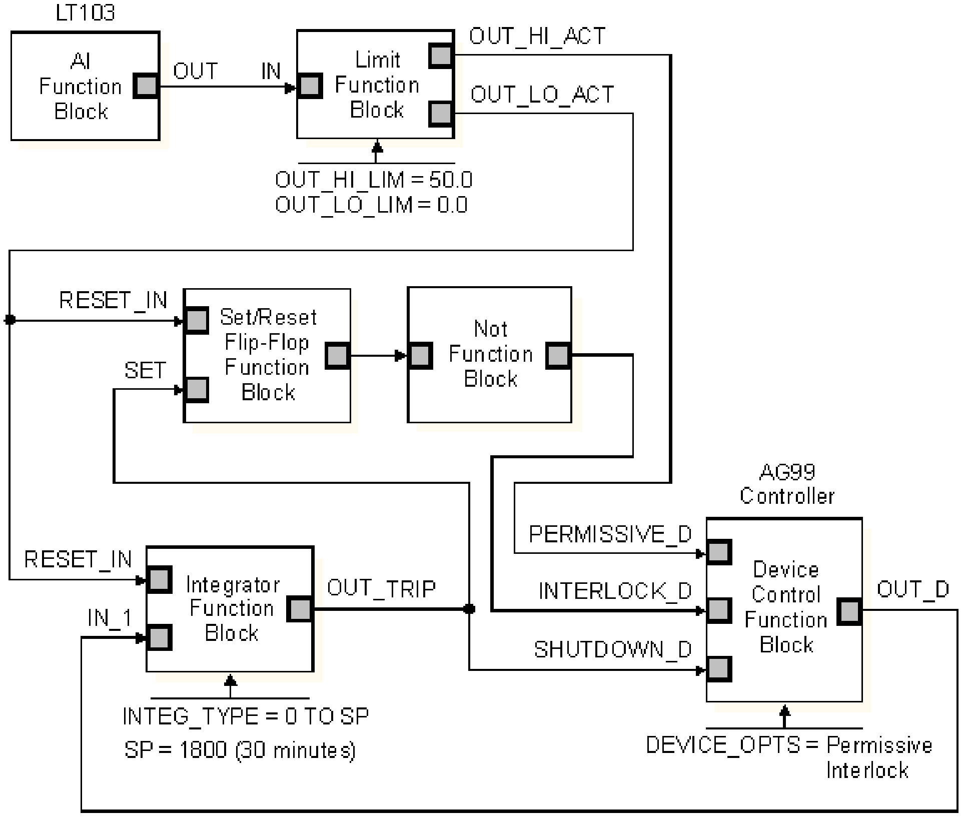
Warning Disabling this option can potentially start the equipment when unattended if the SP is left Active following a failed start due to an interlock.

The following figure shows an example of this mixing process:



Device Control Function Block Agitator Application Example

The following figure shows an example function block diagram for this application.



Device Control Function Block Diagram for Agitator Example

The Device Control function block's PERMISSIVE_D input is used to lock out operator agitator activation until the appropriate tank level is reached. The SHUTDOWN_D input is used to automatically shut off the agitator after the required mixing time has passed. The INTERLOCK_D input is used to lock out operator re-activation of the agitator until the tank mixture is drained. The Permissive and Interlock device options must be enabled.

The Analog Input function block reads the level indication in percent and the Limit function block monitors the level. When the level is greater than or equal to 50.0, the OUT_HI_LIM parameter transitions to True and provides the permissive that allows the Device Control block to be commanded to its SLOW or FAST state. When the tank is emptied, the Limit block's OUT_LO_LIM parameter is used to reset the integrator in preparation for the next mixing cycle. OUT_LO_LIM is also used to reset the latched mixing cycle completed signal (OUT_TRIP) generated by the Integrator function block.

After the tank is filled to the desired level, the permissive is satisfied and the operator can command the AG99 Controller to the SLOW or FAST state. The requested state is also the value written to the Integrator block, which is used to compute the mixing time based on the mixing speed. When the Integrator block reaches setpoint (after 15 to 30 minutes, based on the manually selected agitator speeds), its OUT_TRIP parameter is used to shutdown the AG99 Controller through the SHUTDOWN_D parameter in the Device Control block. This same signal is latched to hold the controller at interlock until the tank is emptied, at which time the latch is reset.

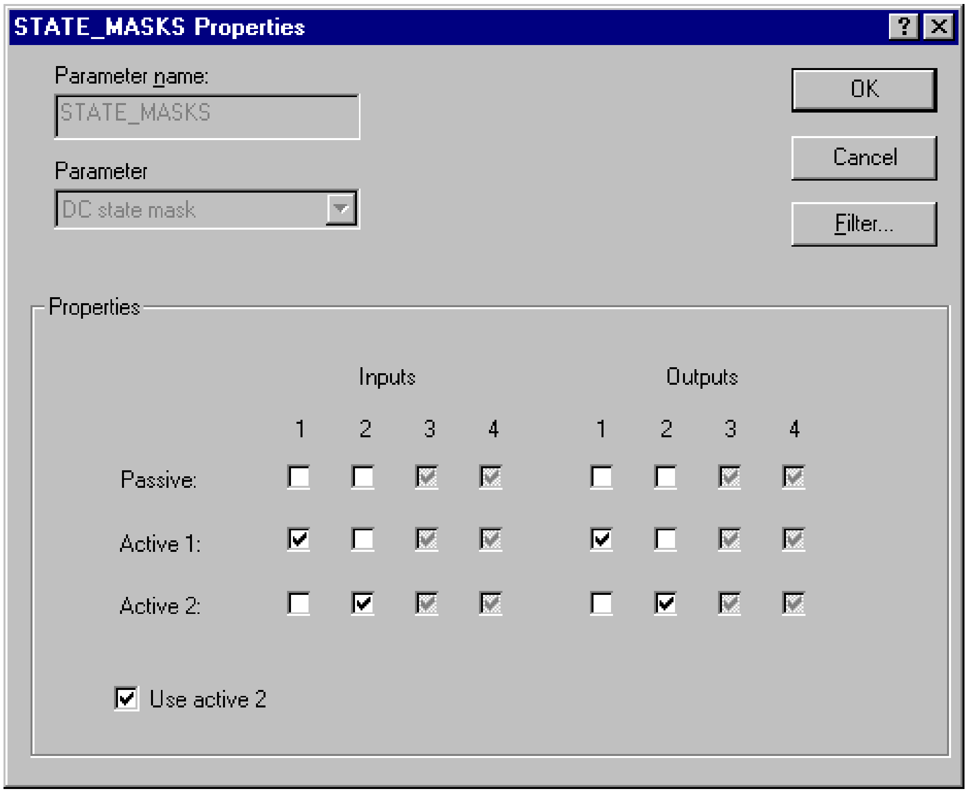
This block logic allows the operator to stop agitation and to restart it in the middle of the mixing cycle without losing elapsed mix time. This is important if the AG99 agitator stops temporarily due to a motor overheating protection switch or a temporary loss of feedback indicators. When the problem is resolved, the agitator can be commanded back to the active mixing speed, picking up where it left off.

The following table shows the discrete I/O combinations used for the three agitator states.

Inputs and Outputs for Device Control Function Block Agitator Example

Agitator Device Status	**Feedback Outputs**		**Command Inputs**	
	First Output	**Second Output**	**First Input**	**Second Input**
OFF	False	False	False	False
SLOW	True	False	True	False
FAST	False	True	False	True

The following figure shows the STATE_MASKS parameter setting used to configure the device controller states for the agitator.



Parameter name: STATE_MASKS

Parameter: DC state mask

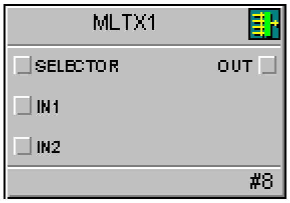
	Inputs				Outputs			
	1	2	3	4	1	2	3	4
Passive:	□	□	☑	☑	□	□	☑	☑
Active 1:	☑	□	☑	☑	☑	□	☑	☑
Active 2:	□	☑	☑	☑	□	☑	☑	☑

☑ Use active 2

Device Control Function Block STATE_MASKS Setting for Agitator Example

Multiplexer Function Block

The Multiplexer (MLTX) function block selects one input value from as many as sixteen input values and places it at the output. The block supports signal status propagation. There are no modes or alarm detection in the Multiplexer function block.



Multiplexer Function Block

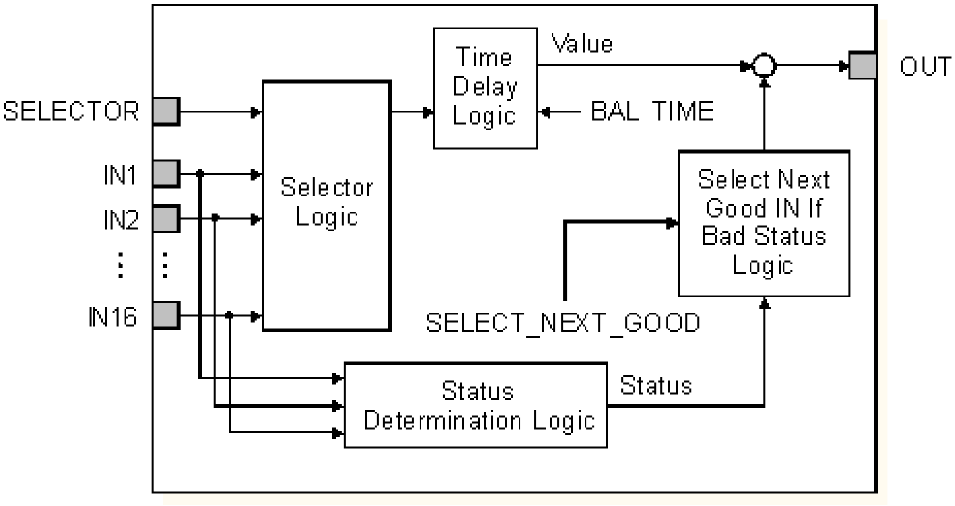
SELECTOR selects the input to place at the output.

IN1 through IN[n] are the analog input values and statuses (as many as 16 inputs).

OUT is the analog output value and status.

Schematic Diagram - Multiplexer Function Block

The following figure shows the internal components of the Multiplexer function block:



Multiplexer Function Block Schematic Diagram

Block Execution - Multiplexer Function Block

The Multiplexer function block reads the values and statuses of as many as sixteen inputs and selects the input designated by the SELECTOR parameter. The number of inputs to the block is an extensible parameter. The block default is two inputs. You add inputs by selecting the function block diagram, clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs. This creates additional input connectors for the block.

The transfer from one input to another can be smoothed by using the balance time parameter (BAL_TIME). The BAL_TIME parameter defines the amount of time for the previous input value to ramp to the newly selected input value. When BAL_TIME is set to zero, the transition between inputs is instantaneous.

You can choose to have the block automatically select the next input with Good status by configuring the SELECT_NEXT_GOOD parameter. Next, if the selected input has Bad status, the next Good input is selected, in ascending order and rotating from last to first. If all of the input statuses are Bad, the block places the selected input at the output and labels it with a Bad status. When this option is False, the block sets the output to the selected input regardless of status.

The following table gives an example of what the output value and status would be with three multiplexed inputs. The input value and status in the table are represented using a value[status] nomenclature. For example, 1[Good] is a value of 1 and a status of Good.

Multiplexer Function Block Response Example

IN1 Value [Status]	IN2 Value [Status]	IN3 Value [Status]	SELECTOR	SELECT_ NEXT_ GOOD	OUT Status	OUT
10 [Good]	20 [Bad]	30 [Good]	1	False	Good	10
10 [Bad]	20 [Good]	30 [Good]	1	True	Good	20
10 [Good]	20 [Bad]	30 [Bad]	2	True	Good	10
10 [Good]	20 [Good]	30 [Good]	2	False	Good	20
10 [Bad]	20 [Bad]	30 [Bad]	3	True	Bad	30

Status Handling - Multiplexer Function Block

The output status is set to the worst status of the selected input signal and the SELECTOR input.

Parameters - Multiplexer Function Block

The following table lists the system parameters for the Multiplexer function block:

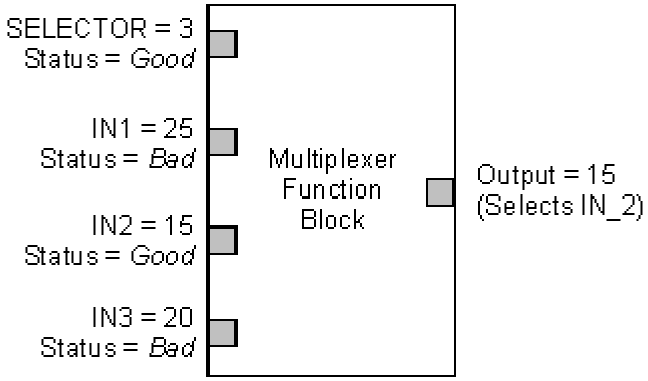
Multiplexer Function Block System Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active) or the indication that an error condition (at the module level) not selected in MERROR_MASK is True (Active) or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active) or the indication that an error condition (at the module level) selected in MERROR_MASK is True (Active) or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions are True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions that are not included in BAD_MASK are True, ABNORM_ACTIVE becomes True.
BAL_TIME	Seconds	Specifies the time for the old input value to ramp to the newly selected input value.
BLOCK_ERR	None	The summary of active error conditions associated with the block. The possible block errors are Block configuration error, Simulate active, Local override, Input failure/process variable has *Bad* status, Output failure, Readback failed, Out of Service, and Other. Each function block reports none or a subset of these error conditions.
IN1 to IN16	Determined by source	The analog input values and statuses. The number of inputs is an extensible parameter.
OUT	Determined by source	The analog output value and status.
SELECT_NEXT_GOOD	None	Selects the next input with Good status when the selected input has Bad status (True [1] = enabled, False [0] = disabled).
SELECTOR	None	Selects the input to place at the output.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Multiplexer Function Block

The Multiplexer function block selects one input out of a number of inputs. You select the input by operator action or by logic from another function block. You can use the Multiplexer function block to select one out of several transmitter values as shown in the following figure:



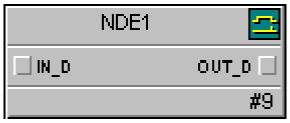
Multiplexer Function Block Diagram Example

In this example, the SELECT_NEXT_GOOD parameter is set to True. IN 1 is the value of transmitter A and IN2 is the value of transmitter B. IN3 is the average of the two. Normally, the output (OUT) would be the value of IN3 in this case, because SELECTOR equals 3. Suppose transmitter A fails, causing the status of IN1 and IN3 to be Bad. Because SELECT_NEXT_GOOD is True, the selected output will be the value of IN2.

Negative Edge Trigger Function Block

The Negative Edge Trigger (NDE) function block generates a True (1) discrete pulse output when the discrete input makes a negative (True-to-False) transition since the last execution of the block. If there has been no transition, the discrete output of the block is False (0).

The Negative Edge Trigger function block supports signal status propagation. There are no modes or alarm detection in the block.



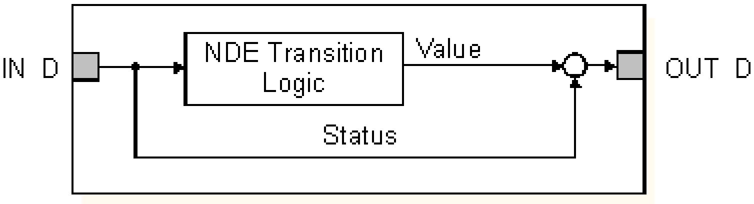
Negative Edge Trigger Function Block

IN_D is the discrete input value and status.

OUT_D is the discrete output value and status.

Schematic Diagram - Negative Edge Trigger Function Block

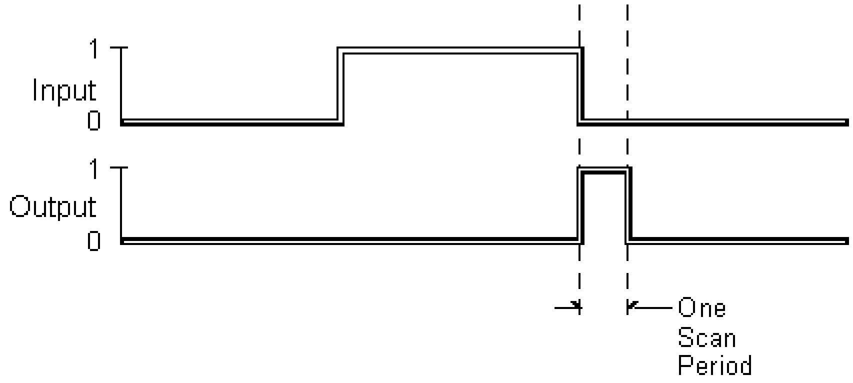
The following figure shows the internal components of the Negative Edge Trigger function block:



Negative Edge Trigger Function Block Schematic Diagram

Block Execution - Negative Edge Trigger Function Block

The Negative Edge Trigger function block is used to trigger other logical events based on the falling transition of a logical signal. If the input value has changed from True to False since the block was last executed, the output of the block is set True. If the value has not changed from True to False, the block output is set False. The following figure shows how the Negative Edge Trigger function block responds to a change in input:



Negative Edge Trigger Function Block Execution Example

Status Handling - Negative Edge Trigger Function Block

The output status is set to the input status.

Parameters - Negative Edge Trigger Function Block

The following table lists the system parameters for the Negative Edge Trigger function block:

Negative Edge Trigger Function Block System Parameters

Parameter	Units	Description
IN_D	None	The discrete input value and status.
OUT_D	None	The discrete output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Negative Edge Trigger Function Block

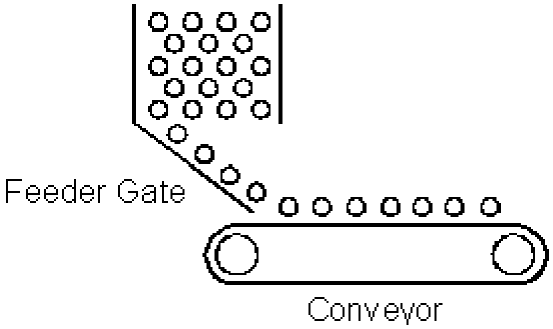
The Negative Edge Trigger function block is used to trigger other logical events based on the falling transition of a logical signal. For example, you can use the Negative Edge Trigger block to trigger actions based on the shutdown of equipment.

Note The output of this block should not be used in a different module unless you are certain that the scan rates match and will not be changed in the future.

You can lengthen the pulse by using an Off-delay Timer function block, or you can use the pulse output to set a latch.

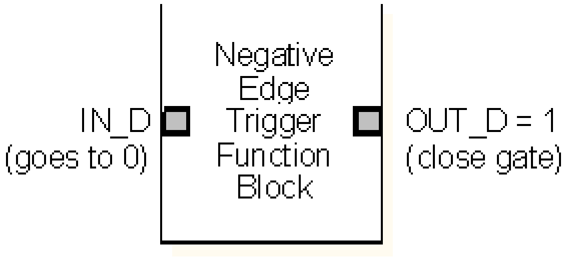
Application Example: Conveyor System

Assume a conveyor feeder gate must be closed when the conveyor shuts down, as shown in the following figure:



Negative Edge Trigger Function Block Application Example

You configure the block to trigger an output that closes the gate when the input (conveyor motor) transitions to False (0). The following figure is the function block diagram for this example:

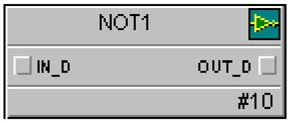


Negative Edge Trigger Function Block Diagram Example

Not Function Block

The Not function block logically inverts a discrete input signal and generates a discrete output value. When the input is True (1), the output is False (0). When the input is False, the output is True.

The block supports signal status propagation. There are no modes or alarm detection in the Not function block.



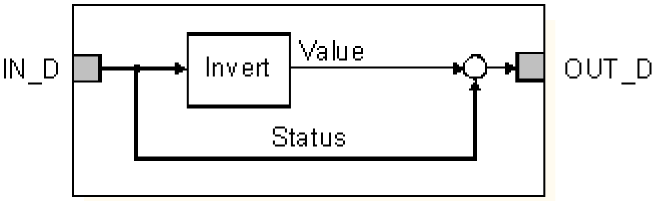
Not Function Block

IN_D is the discrete input value and status.

OUT_D is the discrete output value and status.

Schematic Diagram - Not Function Block

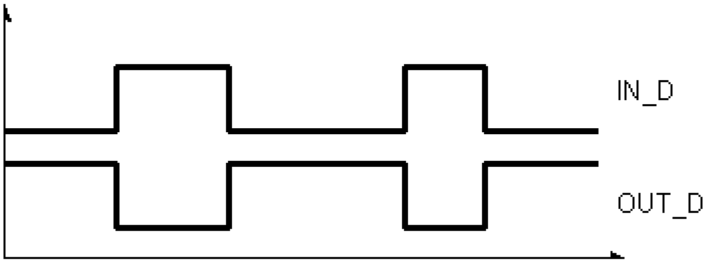
The following figure shows the internal components of the Not function block:



Not Function Block Schematic Diagram

Block Execution - Not Function Block

The Not function block generates an output value that is the logical NOT of its input. When the input is False, the output is True. When the input is True (one or greater), the output is False. The following figure shows an example of Not function block execution.



Not Function Block Execution Example

Status Handling - Not Function Block

The output status is set to the input status.

Parameters - Not Function Block

The following table lists the system parameters for the Not function block:

Not Function Block System Parameters

Parameter	Units	Description
IN_D	None	The discrete input value and status.
OUT_D	None	The discrete output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

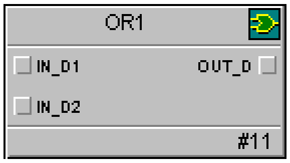
Application Information - Not Function Block

The Not function block is used to perform a logical NOT function on a number of inputs. For example, you can close a normally open valve when a discrete input signal is lost or becomes False.

Or Function Block

The Or function block generates a discrete output value based on the logical OR of two to sixteen discrete inputs. When one or more of the inputs is True (1), the output is set to True.

The block supports signal status propagation. There are no modes or alarm detection in the Or function block.



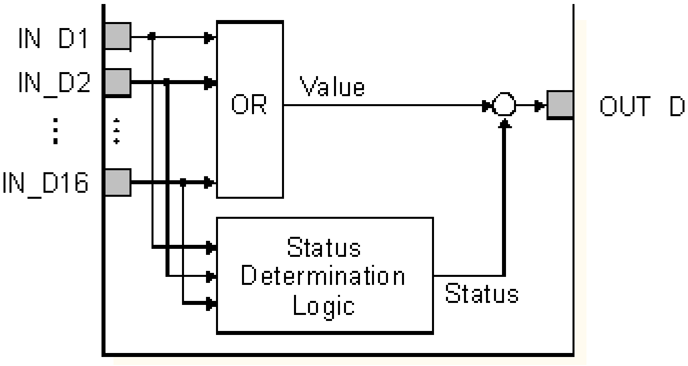
Or Function Block

IN_D1 through IN_D[n] are the discrete input values and statuses (as many as 16 inputs).

OUT_D is the discrete output value and status.

Schematic Diagram - Or Function Block

The following figure shows the internal components of the Or function block:



Or Function Block Schematic Diagram

Block Execution - Or Function Block

The number of inputs to the Or function block is an extensible parameter. The block default is two inputs. You add inputs by selecting the function block diagram, clicking the right mouse button, clicking Extensible Parameters, and modifying the number of inputs. This creates additional input connectors for the block. When one or more of the inputs is True (1), the output is set to True. Otherwise, the output is set to False.

Status Handling - Or Function Block

The output status is set to the worst among the input statuses. However, when at least one input is True and its status is not Bad, the output status is set to Good: Process.

Parameters - Or Function Block

The following table lists the system parameters for the Or function block:

Or Function Block System Parameters

Parameter	Units	Description
IN_D1 to IN_D16	None	The discrete input value and status. The number of inputs is an extensible parameter.
OUT_D	None	The discrete output value and status.

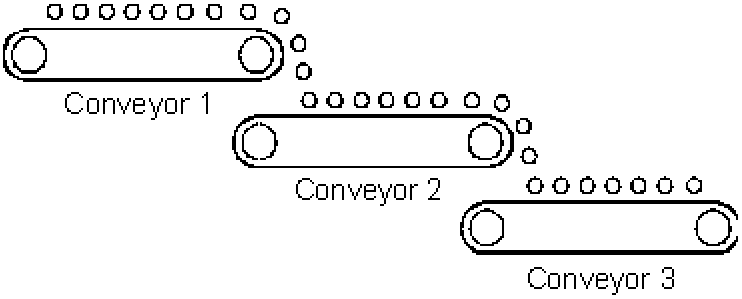
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Or Function Block

The Or Function Block is used to perform a logical OR function on a number of discrete inputs. You can use the Or block to trigger events based on equipment status.

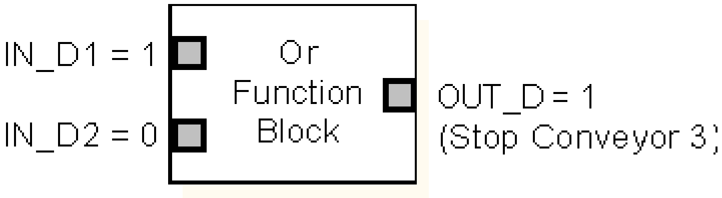
Application Example: Equipment Shutdown

Assume a downstream conveyor must be shut down when one of the upstream conveyors stops. The following figure shows an example conveyor system:



Or Function Block Application Example

You configure the Or function block to stop Conveyor 3 when the inputs from Conveyor 1 or Conveyor 2 indicate a shutdown. The following figure is the function block diagram for this example:

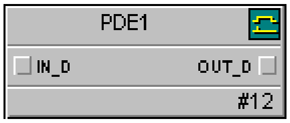


Or Function Block Diagram Example

Positive Edge Trigger Function Block

The Positive Edge Trigger (PDE) function block generates a True (1) discrete pulse output when the discrete input makes a positive (False-to-True) transition since the last execution of the block. If there has been no transition, the discrete output of the block is False (0).

The Positive Edge Trigger function block supports signal status propagation. There are no modes or alarm detection in the block.



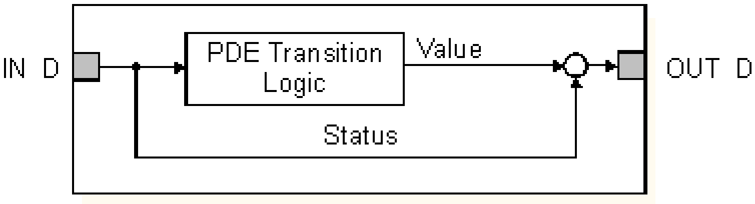
Positive Edge Trigger Function Block

IN_D is the discrete input value and status.

OUT_D is the discrete output value and status.

Schematic Diagram - Positive Edge Trigger Function Block

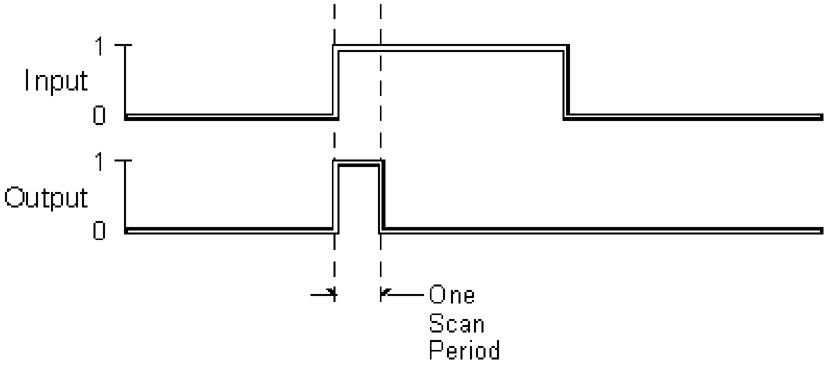
The following figure shows the internal components of the Positive Edge Trigger function block:



Positive Edge Trigger Function Block Schematic Diagram

Block Execution - Positive Edge Trigger Function Block

The Positive Edge Trigger function block is used to trigger other logical events based on the rising transition of a logical signal. If the input value has changed from False to True since the block was last executed, the output of the block is set True. Otherwise, the output is False. The following drawing shows how the Positive Edge Trigger function block responds to a change in input:



Positive Edge Trigger Function Block Execution Example

Status Handling - Positive Edge Trigger Function Block

The output status is set to the input status.

Parameters - Positive Edge Trigger Function Block

The following table lists the system parameters for the Positive Edge Trigger function block:

Positive Edge Trigger Function Block System Parameters

Parameter	Units	Description
IN_D	None	The discrete input value and status.
OUT_D	None	The discrete output value and status.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Positive Edge Trigger Function Block

You use the Positive Edge Trigger function block to trigger events based on the rising transition of a logical signal. For example, you can use the block to trigger a machine startup when a valve opens.

Note The output of this block should not be used in a different module unless you are certain that the scan rates match and will not be changed in the future.

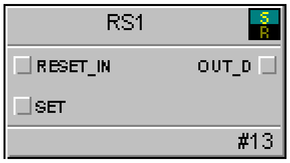
You can lengthen the pulse by using an Off-delay Timer function block, or you can use the pulse output to set a latch.

Reset/Set Flip-flop Function Block

The Reset/Set Flip-flop (RS) function block generates a discrete output value based on NOR logic of reset and set inputs:

- If the reset input is False (0) and the set input is True (1), the output is True. The output remains True, regardless of the set value, until the reset value is True. When reset becomes True, the output is False.
- When both inputs are True, the output is False.
- When both inputs become False, the output remains at its last state and can be either True or False.

There are no modes or alarm detection in the block.



Reset/Set Flip-flop Function Block

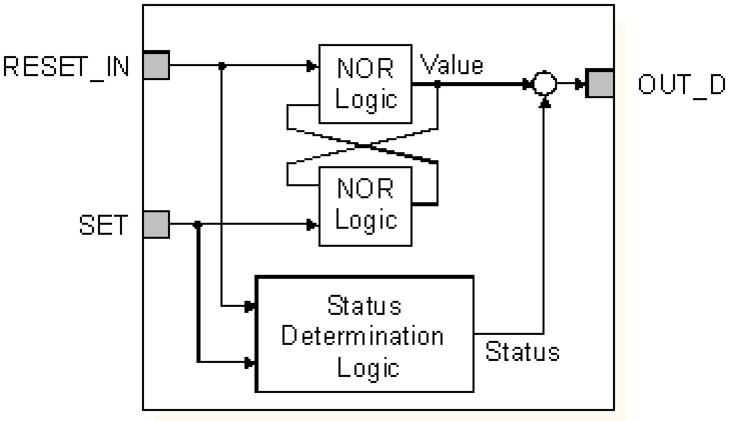
RESET_IN is the reset discrete input value and status.

SET is the set discrete input value and status.

OUT_D is the discrete output value and status.

Schematic Diagram - Reset/Set Flip-flop Function Block

The following figure shows the internal components of the Reset/Set Flip-flop function block:



Reset/Set Flip-flop Function Block Schematic Diagram

Block Execution - Reset/Set Flip-flop Function Block

The Reset/Set Flip-flop function block is used to detect when the set input (SET) transitions to True. It holds the output True, even when SET transitions to False, until another event changes the reset input (RESET_IN) to True.

The following table shows the block output value based on the possible SET and RESET_IN combinations:

Reset/Set Flip-flop Function Block Truth Table

SET	RESET_IN	OUT_D
False	False	Last OUT
False	True	False
True	False	True
True	True	False

Status Handling - Reset/Set Flip-flop Function Block

The output status is equal to the worst status among the inputs.

Parameters - Reset/Set Flip-flop Function Block

The following table lists the system parameters for the Reset/Set Flip-flop function block:

Reset/Set Flip-flop Function Block System Parameters

Parameter	Units	Description
OUT_D	None	The discrete output value and status.
RESET_IN	None	The reset discrete input value and status used in block logic.
SET	None	The set discrete input value and status used in block logic.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Reset/Set Flip-flop Function Block

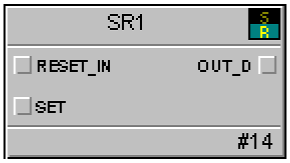
You can use the Reset/Set Flip-flop function block to activate a process that requires a continuous True condition using a momentary switch to start the process.

Set/Reset Flip-flop Function Block

The Set/Reset Flip-flop (SR) function block generates a discrete output value based on NAND logic of set and reset inputs:

- When the reset input is False (0) and the set input is True (1), the output is True. The output remains True until the reset input is True and the set input is False.
- When the reset input is True, the output is equal to the set input.
- When both inputs are True, the output is True.
- When both inputs become False, the output remains at its last state and can be either True or False.

There are no modes or alarm detection in the block.



Set/Reset Flip-flop Function Block

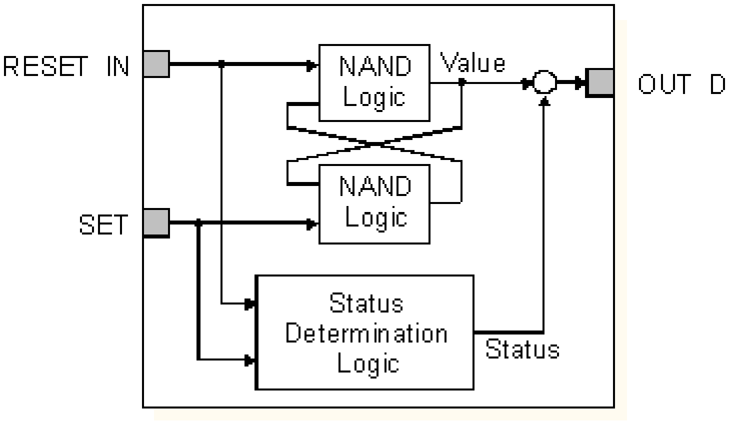
RESET_IN is the reset discrete input value and status.

SET is the set discrete input value and status.

OUT_D is the discrete output signal and status.

Schematic Diagram - Set/Reset Flip-flop Function Block

The following figure shows the internal components of the Set/Reset Flip-flop function block:



Set/Reset Flip-flop Function Block Schematic Diagram

Block Execution - Set/Reset Flip-flop Function Block

The Set/Reset Flip-flop function block is used to detect a change in the set input (SET). When the reset input (RESET_IN) is False, OUT is set True after SET changes to True. OUT remains True, even when SET returns to False, and remains True until RESET_IN is changed to True and SET is False.

The following table shows the block output value based on the possible SET and RESET_IN combinations:

Set/Reset Flip-flop Function Block Output Values

SET	RESET_IN	OUT
False	False	Last OUT
False	True	False
True	False	True
True	True	True

Status Handling - Set/Reset Flip-flop Function Block

The output status is equal to the worst status among the inputs.

Parameters - Set/Reset Flip-flop Function Block

The following table lists the system parameters for the Set/Reset Flip-flop function block:

Set/Reset Flip-flop Function Block System Parameters

Parameter	Units	Description
OUT_D	None	The discrete output value and status.
RESET_IN	None	The reset discrete input value and status used in block logic.
SET	None	The set discrete input value and status used in block logic.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

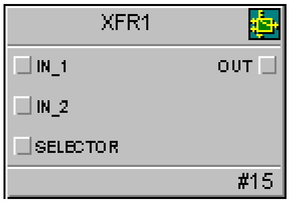
Application Information - Set/Reset Flip-flop Function Block

You can use the Set/Reset Flip-flop function block to activate a process that requires a continuous True condition using a momentary switch to start the process.

Transfer Function Block

The Transfer (XFR) function block selects one of two analog input signals and transfers the selected input to the output after a specified time. The transfer from one input to another is smoothed with a linear ramp.

The Transfer function block supports signal status propagation. There are no modes or alarm detection in the block.



Transfer Function Block

IN_1 is the first analog input and status.

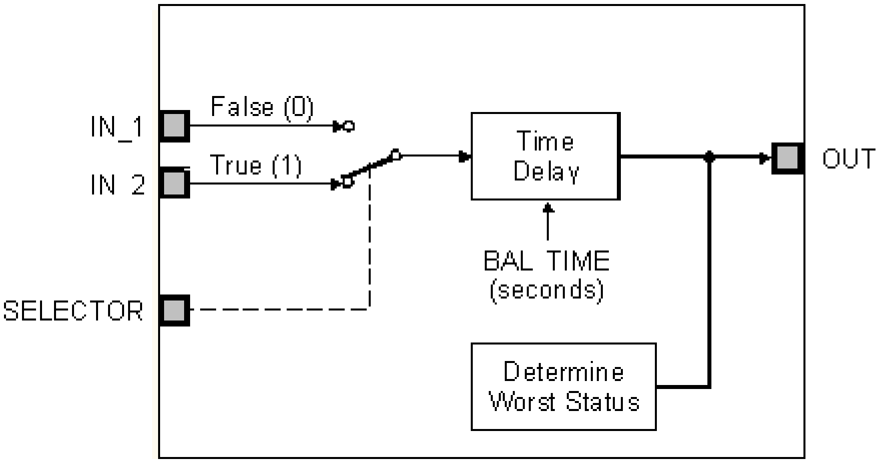
IN_2 is the second analog input and status.

SELECTOR selects the input to be placed at the output.

OUT is the analog output value and status.

Schematic Diagram - Transfer Function Block

The following figure shows the internal components of the Transfer function block:



Transfer Function Block Schematic Diagram

Block Execution - Transfer Function Block

The Transfer function block selects one of two inputs based on the SELECTOR parameter value. When SELECTOR is False (0), IN_1 is transferred to the output after a specified time (BAL_TIME). When SELECTOR is True (1), IN_2 is transferred to the output after BAL_TIME. When BAL_TIME = 0, the output changes instantaneously to the new value.

Status Handling - Transfer Function Block

The output status is set to the worse status of the selected input and SELECTOR.

Parameters - Transfer Function Block

The following table lists the system parameters for the Transfer function block:

Transfer Function Block System Parameters

Parameter	Units	Description
BAL_TIME	Seconds	Specifies the time for the output value to ramp to the newly selected output value.
IN_1	Determined by source	The first analog input value and status.
IN_2	Determined by source	The second analog input value and status.
OUT	EU of IN	The analog output value and status.
SELECTOR	None	Selects the input to place at the output.

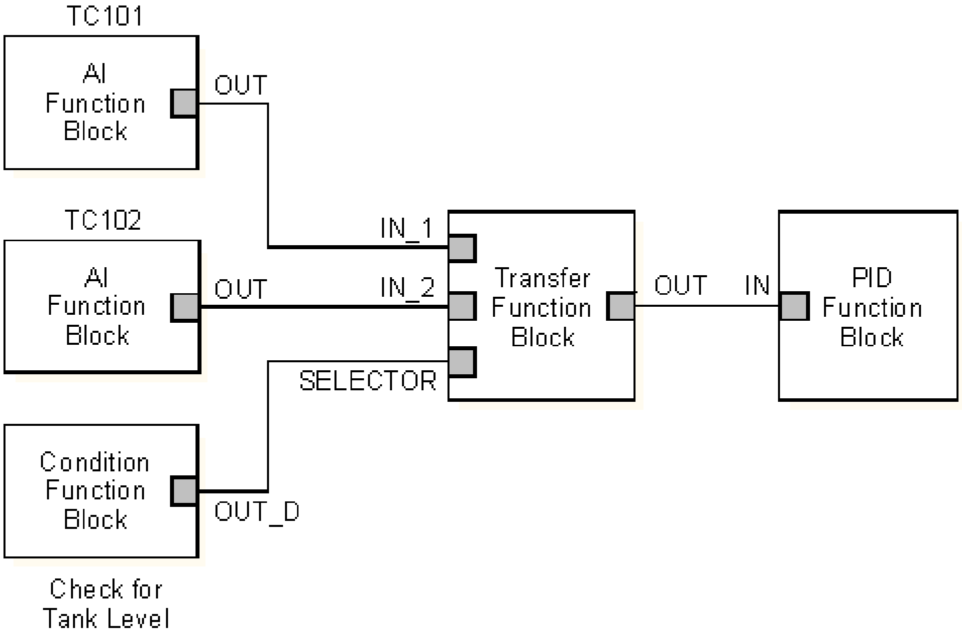
Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Transfer Function Block

You can use the Transfer function block to prevent sudden changes from being sent to the process. For example, if your process cannot handle a sudden change in flow, the Transfer function can ramp the flow to the new value. You can also use the block to transfer control to a predetermined value when a field value status goes bad.

Application Example: Temperature Sensor Switch

You can use the Transfer function block as an input switch between two temperature sensors. The SELECTOR parameter selects the temperature input based on the level in a tank. The following figure is an example function block diagram for this application:



Transfer Function Block Diagram Example

Advanced Control Blocks

This chapter contains information on advanced control function blocks in the DeltaV system.

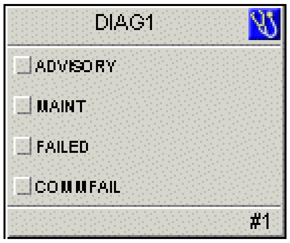
Diagnostic (DIAG) Function Block

The Diagnostic function block provides a method to monitor device alerts from non-fieldbus assets. In modules, wire parameters or block outputs that indicate device health for non-fieldbus blocks to the Diagnostic block. The alerts from these diagnostic blocks are monitored by the Inspect application. The block sets the output (STATUS_INDEX) value based on the values of the inputs. The block does not support modes. The inputs have the following values when true:

- ADVISORY = 0x1000
- COMMFAIL = 0x2000
- FAILED = 0x4000
- MAINT = 0x8000

Note: Each DIAG function block must have a unique name in the entire DeltaV system. It is recommended that you name the blocks for the asset they refer to. The unique names are important because the Asset page of the Inspect application lists only the DIAG block name. If two or more DIAG blocks have the same name the information in Inspect will be misleading because operators cannot determine which block's information is being displayed.

Note that fieldbus blocks are monitored automatically and do not require Diagnostic blocks.



Diagnostic Function Block

Block Execution - Diagnostic Function Block

The block sets the bits of output based on the values of the input. The inputs have the following values when true:

- ADVISORY = 0x1000
- COMMFAIL = 0x2000
- FAILED = 0x4000
- MAINT = 0x8000

The value of the output (STATUS_INDEX) is the sum of the values for the inputs that are true.

Parameters - Diagnostic Function Block

The following table lists the system parameters for the Diagnostic function block:

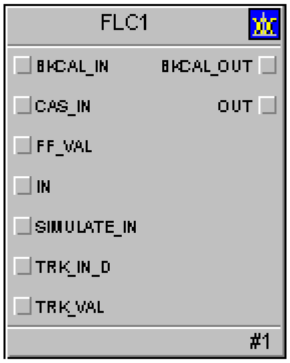
Diagnostic Function Block Parameters

Parameter	Units	Description
ADVISORY	None	True = the alert is active False = the alert is not active
COMMFAIL	None	True = the alert is active False = the alert is not active
FAILED	None	True = the alert is active False = the alert is not active
MAINT	None	True = the alert is active False = the alert is not active
STATUS_INDEX	None	A numerical value that indicates which alerts are currently active.

Fuzzy Logic Control (FLC) Function Block

The Fuzzy Logic Control (FLC) function block provides the control capability of the PID block with the added benefit of superior response for both setpoint changes and external load disturbances. By using fuzzy logic, the FLC function block minimizes overshoot and provides good load disturbance rejection. The scaling factors of the FLC function block can be automatically established using DeltaV Tune.

The FLC function block operates by using predefined fuzzy rules, membership functions, and adjustable parameters known as scaling factors. The FLC function block translates the loop's absolute values into fuzzy values by calculating the scaled error (*e*) and scaled change in error (D*e*) in addition to the degree of membership in each of the predefined membership functions. It then applies the fuzzy rules and, finally, retranslates the values into a control move.



Fuzzy Logic Control Function Block

This function block supports mode control, signal scaling and limiting, feedforward control, override tracking, alarm limit detection, and signal status propagation.

In Cascade (Cas) mode, the setpoint (SP) is adjusted by a master controller. In Remote Cascade mode (RCas), the setpoint is written by an application. In Automatic (Auto) mode, the SP can be adjusted by the operator. In RCas, Cas and Auto modes, the output is calculated to maintain setpoint. In Manual (Man) mode, the block's output is set by the operator. In Remote Output (ROut) mode, the block's output is written by an application.

The FLC function block can be connected directly to process I/O, can receive its input from another block at the input connection (IN), or can provide an output value to another block through the output connection (OUT).

You can connect BKCAL_OUT to a master controller and BKCAL_IN to a slave controller to compensate for downstream limits and to provide bumpless transfer to closed loop control.

You connect the tracking inputs (TRK_IN_D and TRK_VAL) for externally controlled output tracking.

BKCAL_IN is the analog input value and status from another block's BKCAL_OUT output that is used for backward output tracking for bumpless transfer and to pass limit status.

CAS_IN is the remote setpoint (SP) value from another block.

FF_VAL is the feedforward control input value and status.

IN is the connection for the process variable (PV) from another function block.

SIMULATE_IN is the input value and status used by the block instead of the analog measurement when simulation is enabled.

TRK_IN_D initiates the external tracking function.

TRK_VAL is the value after scaling applied to OUT in Local Override mode.

BKCAL_OUT is the value and status required by the BKCAL_IN input of another block to compensate for downstream limits and to provide bumpless transfer to closed loop control.

OUT is the block output value and status.

Other FLC Function Block Features

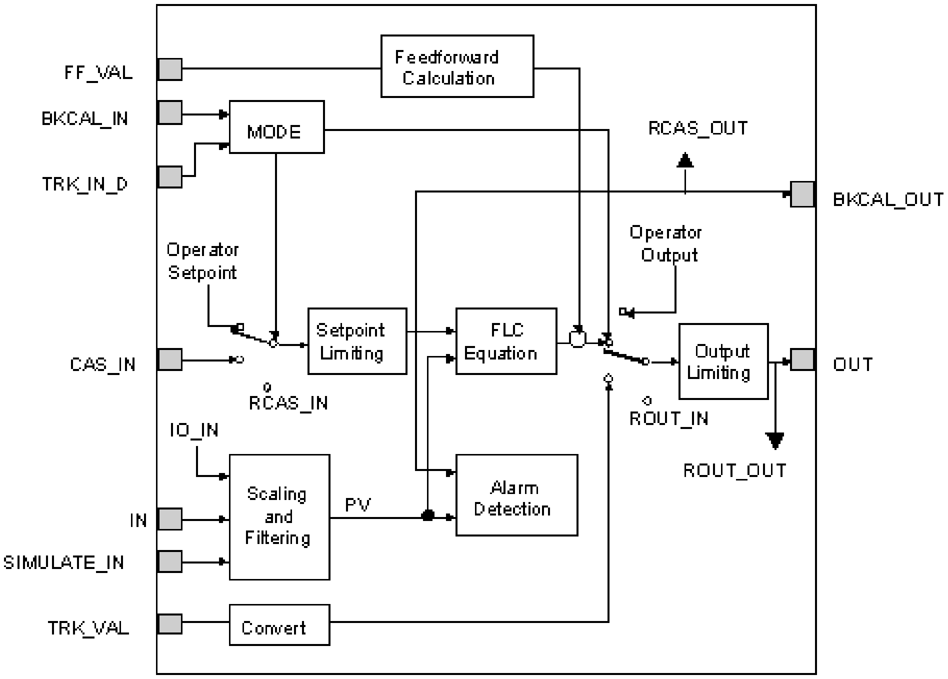
Many of the following features of the FLC function block are identical to those provided for the PID function block.

- alarm detection
- status handling
- I/O Selection
- simulation
- signal Conversion
- filtering
- feedforward Calculation
- tracking
- setpoint selection and limiting
- output selection and limiting
- bumpless transfer and setpoint tracking
- reverse and direct action
- block errors

Refer to the PID function block topic for details about these features.

Schematic Diagram - Fuzzy Logic Control Function Block

The following diagram shows the internal components of the Fuzzy Logic Control function block:



Fuzzy Logic Control Function Block Schematic Diagram

Block Execution - Fuzzy Logic Control Function Block

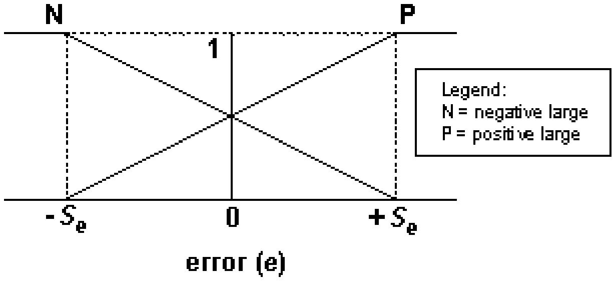
The nonlinearity built into the FLC function block reduces overshoot and settling time, achieving tighter control of the process loop. Specifically, the FLC function block treats small control errors differently from large control errors and penalizes large overshoots more severely. It also severely penalizes large changes in the error, helping to reduce oscillation.

This section describes how the Fuzzy Logic Control block functions.

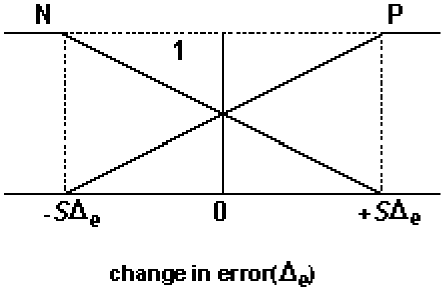
Two Membership Functions

The FLC function block uses two membership functions: the input signals are error and change in error, and the output signal is the change in control action. The relations among these three variables represent a nonlinear controller. The nonlinearity results from a translation of process variables to a fuzzy set (fuzzification), rule inference, and retranslation of a fuzzy set to a continuous signal (defuzzification).

The two membership functions for error, change in error and change in output are **negative** and **positive**. The membership scaling (*Se* and *S* D*e*) and the error value and change in error, respectively, determine the degree of membership.

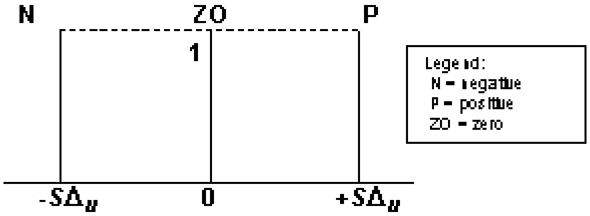


Error Membership Functions



Change in Error Membership Functions

The change in output membership functions are called **singletons**. They represent fuzzy sets whose support is a single point with a membership function of one. The membership scaling (*S* D*u*) determines the magnitude of output change for a given error and change in error.



Change in Output Singleton Membership Functions

Four Fuzzy Logic Rules

There are four fuzzy logic rules that the FLC function block uses for a reverse acting controller.

DeltaV Fuzzy Logic Rules

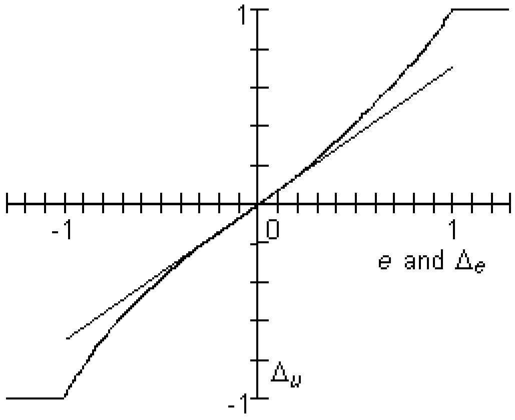
Number	Rule
Rule 1	If error is N and change in error is N, make change in output P.
Rule 2	If error is N and change in error is P, make change in output ZO.
Rule 3	If error is P and change in error is N, make change in output ZO.
Rule 4	If error is P and change in error is P, make change in output N.

Refer to the Fuzzy Logic Evaluation topic for an explanation and example of how the fuzzy membership functions and rules are used in fuzzy logic evaluation.

Fuzzy Logic Control Nonlinear PI Relationship

The two membership functions associated with each input variable and three membership functions for the output variable makes the FLC function block nonlinear in its response.

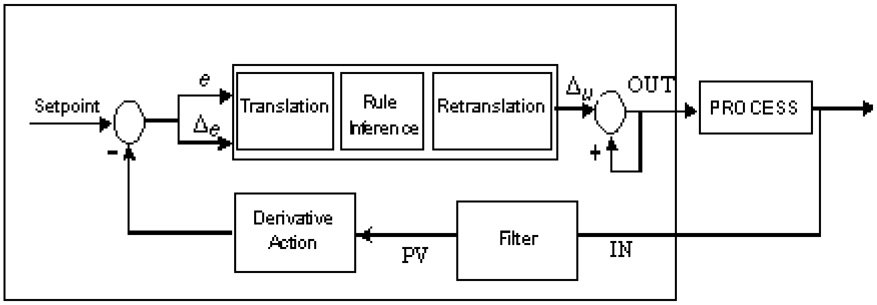
For regions where the absolute error is greater than the error scaling factor or the absolute change in error is greater than the change in error scaling factor, the values for error and change in error are clipped at the error scaling factor and change in error scaling factor, respectively. The following figure shows an example FLC curve that illustrates how the change in controller gain is smooth and continuous using only two input membership functions and three output membership functions.



FLC Function Block's Nonlinear Relationship

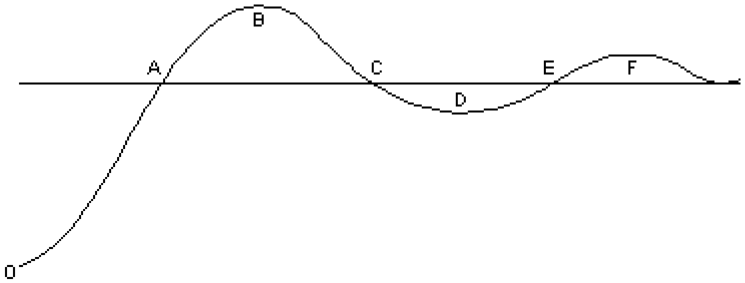
The dark line shows the FLC function block's nonlinear relationship when the error is equal to the change in error. The straight line through the origin shows the linear relationship of a standard PI controller. As the error and change in error increase, the change in output of a standard PI controller increases linearly. Note that the gain of the FLC function block is similar to the gain of the PI controller when the error and change in error are small. The gain of the FLC function block increases gradually as the error and change in error increase.

The nonlinearity built into the FLC function block reduces overshoot and settling time, achieving tighter control of the process loop. To help anticipate a rapid change in the process with the FLC function block, derivative action is provided in the feedback path of the loop, as shown in the following figure.



Fuzzy Logic Control with Derivative Action

The FLC function block treats small control errors differently from large control errors and penalizes large overshoots more severely. It also severely penalizes large changes in the error, helping to reduce the oscillation.



Process Variable Oscillation Example

The above figure depicts how an FLC function block reacts to overshoot and oscillation. At points B, D, and F, where overshoot occurs, the FLC function block applies stronger control actions to bring the variable back to the setpoint. At points A, C, and E, where large changes in error occur and are dominant, the FLC function block applies stronger corrective actions to reduce oscillation.

This type of nonlinearity allows the FLC function block to provide better control performance than standard PID control.

Establishing Scaling Factors

DeltaV Tune can be used to establish the scaling factors (*Se*, *S*D*e*, and *S*D*u*). For a small control error and setpoint change less than a nominal value (D*Ysp*), the FLC function block scaling factors are related to the proportional gain (Kp) and reset (Ti), which would be used in a PI block executing at a one (1) second scan rate (D*t*) to control the same process. Refer to the following equations:

$$S\Delta_e = \beta\Delta Y_{sp}$$

$$S\Delta_u = 2S\Delta_e Kp$$

$$S_e = S_{e0} = TiS\Delta_e$$

where:

*S*D*e* = change of error scaling

Se = error scaling

S D*u* = change of controller output scaling

Se0 = error scaling for a one (1) second scan rate

Beta is a function of process deadtime (DT) and ultimate period or time constant (TC) and has values in the following range:

$$.2 < \beta < .5$$

The approximate formula for calculating beta is as follows:

$$\beta = .2 + \frac{DT}{TC}$$

The Fuzzy Logic Control function block accounts for the scan rate and recalculates the error scaling factor (*Se*), which depends on the scan rate appropriate to the function block scan (D*t*).

$$S_e = \frac{S_{eo}}{\Delta t} = \frac{TiS\Delta_e}{\Delta t}$$

The Fuzzy Logic Control function block is designed to be set up by DeltaV Tune. However, if you choose to set up scaling factors manually, it is important to note that, in such a case, DeltaV Tune will **not** set up derivative time for the FLC block, and you will **not** have manual access to the derivative term. In such a situation, the FLC block, effectively, becomes the PI controller. This can affect block performance significantly.

Note The nominal setpoint change value for D*Ysp* is one percent.

When the setpoint change is greater than the nominal setpoint change (D*Ysp*), these scaling factors are internally increased by the Fuzzy Logic Control function block. This internal scaling is changed in the ratio of actual setpoint change to the nominal setpoint change. These larger scaling factors are used while the control error (PV–SP) remains large due to the change in setpoint. When the control error has returned to a small value and remains small for a period of time, the scaling factors used by the fuzzy algorithm are once again the block scaling parameter values.

Modes - Fuzzy Logic Control Function Block

The Fuzzy Logic Control function block supports the following modes:

Manual (Man) – The block output (OUT) can be set manually.

Remote Output (ROut) – The OUT is calculated by an application that writes to the ROUT_IN parameter.

Automatic (Auto) – The SP can be set manually and the block algorithm calculates OUT.

Cascade (Cas) – The SP is calculated in another block and is provided to the Fuzzy Logic Control function block through the CAS_IN connection.

Remote Cascade (RCas) – The SP is calculated by an application that writes to the RCAS_IN parameter.

Local Override (LO) – The track function is active. OUT is set by TRK_VAL. The BLOCK_ERR parameter shows Local override.

Initialization Manual (IMan) – The output path is not complete (for example, the cascade-to-slave path might not be open). In IMan mode, OUT tracks BKCAL_IN.

Out of Service (OOS) – The block is not processed. The OUT status is set to Bad: Out of Service. The BLOCK_ERR parameter shows Out of service.

You can configure the Man, Auto, Cas, RCas, ROut and OOS modes as permitted modes for operator entry.

Parameters - Fuzzy Logic Control Function Block

The following table lists the system parameters for the Fuzzy Logic Control function block:

Fuzzy Logic Control Function Block Parameters

Parameter	Units	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active), an error condition (at the module level) not selected in MERROR_MASK is True (Active), or a module status not selected in MSTATUS_MASK is True (Active).
ALARM_HYS	Percent	The amount that the alarm value must return within the alarm limit before the associated active alarm condition clears. ALARM_HYS is limited to 50% of scale.
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active), an error condition (at the module level) selected in MERROR_MASK is True (Active), or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions is True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions not included in BAD_MASK is True, ABNORM_ACTIVE becomes True.
BKCAL_IN	EU of OUT_SCALE	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer.
BKCAL_OUT	EU of PV_SCALE	The value and status required by the BKCAL_IN input of another block to prevent reset windup and provide bumpless transfer to closed loop control.

Parameter	Units	Description
BLOCK_ERR	None	The summary of active error conditions associated with the block. The block errors for the FLC function block are: Out of Service, Readback Failed, Output Failure, Bad PV, Local Override, Simulate Active, Configuration Error, and Other Error.
BYPASS	None	When enabled and the block is in AUTO mode, bypasses the normal control algorithm by transferring the SP value (in percent) to OUT. When disabled, the block operates normally. To turn BYPASS on or off, select the CONTROL_OPTS Bypass Enable option and set the block to MAN mode.
CAS_IN	EU of PV_SCALE	The remote setpoint value from another block.
CONTROL_OPTS	None	Control options. Allow you to specify control strategy options. The supported control options for the FLC function block are: Use PV for BKCAL_OUT, Track in Manual, Track Enable, Direct Acting, SP-PV Track in LO or IMan, SP-PV Track in Man, and Bypass Enable.
DV_HI_ACT	None	The result of alarm detection associated with DV_HI_LIM. If DV_HI_ACT equals True, DV_HI_LIM has been exceeded.
DV_HI_LIM	EU of PV_SCALE	The amount by which PV can deviate above SP before the deviation high alarm is triggered. When this limit is exceeded, DV_HI_ACT is set to True.
DV_LO_ACT	None	The result of alarm detection associated with DV_LO_LIM. If DV_LO_ACT equals True, DV_LO_LIM has been exceeded.
DV_LO_LIM	EU of PV_SCALE	The amount by which PV can deviate below SP before the deviation low alarm is triggered. When this limit is exceeded, DV_LO_ACT is set to True. Note that DEV_LO_LIM is a negative number and is compared against (PV - SP).
ERROR	EU of PV_SCALE	The difference between SP (setpoint) and PV (process variable).
FF_ENABLE	None	Enables/disables feedforward control.
FF_GAIN	None	The feedfoward gain value. FF_VAL is multiplied by FF_GAIN before it is added to the calculated control output.
FF_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the feedforward value (FF_VAL).
FF_VAL	EU of FF_SCALE	The feedforward control input value and status.
FIELD_VAL	Percent	The value and status from either the I/O card or the simulated input (if simulation is enabled).

Parameter	Units	Description
HI_ACT	None	The result of alarm detection associated with HI_LIM. If HI_ACT equals True, HI_LIM has been exceeded.
HI_HI_ACT	None	The result of alarm detection associated with HI_HI_LIM. If HI_HI_ACT equals True, HI_HI_LIM has been exceeded.
HI_HI_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the high high alarm condition.
HI_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the high alarm condition.
IN	EU of PV_SCALE	The connection for the PV input from another block.
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 if the following conditions are true: - The Write to Inspect Alarm context menu item was selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition will only exist for Variability if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
IO_IN	None	Defines the input DST for the I/O channel used for the PV.
IO_OPTS	None	I/O options. Allow you to select how the I/O signals are processed. The supported I/O options for the FLC function block are: Low Cutoff, Use PV for BKCAL_OUT, Use Fault State Value on Restart, Increase to Close, SP-PV Track in LO or IMan, SP-PV Track in Man, and Invert.
IO_OUT	None	Defines the output DST for the block.
IO_READBACK	None	Defines the Device Signal Tag (DST) for the input channel that provides readback for the value written to the channel defined by IO_OUT.
L_TYPE	None	Linearization type. Determines whether the field value is used directly (Direct), is converted linearly (Indirect), or is converted with the square root (Indirect Square Root).
LO_ACT	None	The result of alarm detection associated with LO_LIM. If LO_ACT equals True, LO_LIM has been exceeded.
LO_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the low alarm condition.
LO_LO_ACT	None	The result of alarm detection associated with LO_LO_LIM. If LO_LO_ACT equals True, LO_LO_LIM has been exceeded.

Parameter	Units	Description
LO_LO_LIM	EU of PV_SCALE	The setting for the alarm limit used to detect the low low alarm condition.
LOW_CUT	EU of PV_SCALE	Activated when the Low Cutoff I/O option is enabled (True). When the converted measurement is below the LOW_CUT value, the PV is set to 0.0.
MODE	None	Parameter used to show and set the block operating state.
OUT	EU of OUT_SCALE	The block output value and status.
OUT_HI_LIM	EU of OUT_SCALE	The maximum output value allowed.
OUT_LO_LIM	EU of OUT_SCALE	The minimum output value allowed.
OUT_READBACK	EU of OUT_SCALE	The value and status of the output channel referenced by IO_READBACK.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
PV	EU of PV_SCALE	The process variable used in block execution and alarm limit detection.
PV_FTIME	Seconds	The time constant of the first-order PV filter.
PV_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with PV.
RCAS_IN	EU of SP_SCALE	The remote analog setpoint value and status. Input provided by either a device or the output of another block.
RCAS_OUT	EU of PV_SCALE	The remote analog setpoint value and status after ramping. Output provided to a device for back calculation and to allow action to be taken under limiting conditions or mode change.
ROUT_IN	EU of OUT_SCALE	Remote output value and status. Input provided by a device to the control block for use as the output (ROut mode).
ROUT_OUT	EU of OUT_SCALE	Remote output value and status. Output provided to a device for back calculation in ROut mode and to allow action to be taken under limiting conditions or mode change.
SF_DELTERR	None	The change-in-error scaling factor. A tuning parameter that converts the change in error (from previous scan) to a normalized value used by the fuzzy logic algorithm.
SF_ERROR	None	The error scaling factor. A tuning parameter that converts the error (PV-SP) to a normalized value used by the fuzzy logic algorithm.

Parameter	Units	Description
SF_OUTPUT	None	The change-in-output scaling factor. A tuning parameter that converts the normalized output of the fuzzy logic algorithm to an increment of change for OUT.
SHED_OPT	None	Defines action to be taken on remote control device timeout.
SHED_TIME	Seconds	The maximum allowable time between RCAS_IN and ROUT_IN updates. If exceeded, mode shedding takes place.
SIMULATE	Percent	Enables simulation and allows you to enter an input value and status. The SIMULATE value is used by the block only when SIMULATE_IN is not connected.
SIMULATE_IN	Percent	The input connector value and status used by the block instead of the analog measurement when simulation is enabled. If SIMULATE_IN is connected or has a manually entered value, SIMULATE_IN will always override SIMULATE. **Note** When SIMULATE_IN is wired from an input source on the function block diagram, it will always override a manually entered value.
SP	EU of PV_SCALE	The block's setpoint value.
SP_FACTOR	Percent	The minimum setpoint change between scans required to initiate scaling factor adaptation.
SP_FTIME	Seconds	Time constant of the first order SP filter.
SP_HI_LIM	EU of PV_SCALE	The highest SP value allowed.
SP_LO_LIM	EU of PV_SCALE	The lowest SP value allowed.
SP_RATE_DN	EU of PV_SCALE per second	Ramp rate at which downward setpoint changes are acted on in Auto mode (in PV units per second). If the ramp rate is set to 0.0, the setpoint will be used immediately. For control blocks, rate limiting applies only in Auto mode. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_RATE_UP	EU of PV_SCALE per second	Ramp rate at which upward setpoint changes are acted on in Auto mode (in PV units per second). If the ramp rate is set to 0.0, the setpoint will be used immediately. For control blocks, rate limiting applies only in Auto mode. For output blocks, rate limiting applies in Auto, Cas, and RCas modes.
SP_WRK	EU of PV_SCALE	The working setpoint of the block. It is the result of setpoint limiting and/or setpoint rate-of-change limiting.
STDEV	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The standard deviation of PV. For analog control blocks in AUTO, mean is assumed to be the SP. Refer to Loop Performance Calculations for more details on how this parameter is calculated.

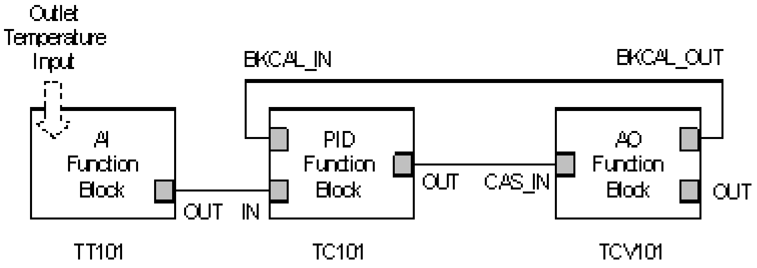
Parameter	Units	Description
STDEV_CAP	EU of OUT_SCALE or EU of PV_SCALE (reports in percent to Inspect)	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.
STATUS_OPTS	None	Status options. Allow you to select options for status handling and processing. The supported status options for the FLC function block are Bad if Limited and Target to Manual if Bad IN.
TRK_IN_D	None	Discrete input that initiates external tracking.
TRK_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with the external tracking value (TRK_VAL).
TRK_VAL	EU of TRK_SCALE	The analog input used in the external tracking function.

Note Default values and data type information for the parameters are available by expanding the Parameter View window.

Application Information - Fuzzy Logic Control Function Block

The Fuzzy Logic Control function block is designed for use in any situation where you need less oscillation and overshoot than is possible with the PID block. Therefore, you can use the FLC function block in any continuous process with significant disturbances or nonlinearities. For example, use the FLC function block to minimize overshoot when fast response to a setpoint or load disturbance is required.

To implement an FLC function block, replace the existing PID with the FLC function block in your control strategy. For example, the blocks and connections required for a single loop using PID and using FLC are shown below:



The FLC and PID Loop

The parameters used for connections when implementing feedforward, cascade or override strategies are the same as for the PID. Refer to the Application Information - PID Function Block topic for examples of such connections.

You can manually adjust the FLC function block scaling to modify the loop response. Also, DeltaV Tune can be used to set the scaling used by the FLC function block automatically. Refer to the DeltaV Tune Overview topic for more information.

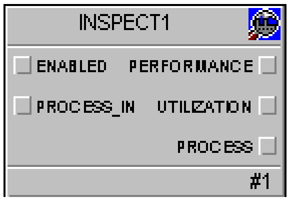
Inspect Function Block

The Inspect function block makes the performance statistics (Performance and Utilization) calculated by the Inspect server available to view, plot, add to history, and so on. If the block's SYSTEM parameter is set to TRUE, the block reports statistics for the entire control system regardless of where it is located. Otherwise, the block reports statistics for the part of the control system where the block is located.

The Inspect server calculates Performance and Utilization values for the specified TIME_FRAME and TIME_WINDOW. The block contains an ENABLED parameter that can be set to 0 or 1 to disable or enable Inspect processing, respectively, on the portion of the plant where the block is located.

Note The AREA parameter of Inspect blocks should be left blank. The parameter exists for backward compatibility with configurations created with earlier versions of the software.

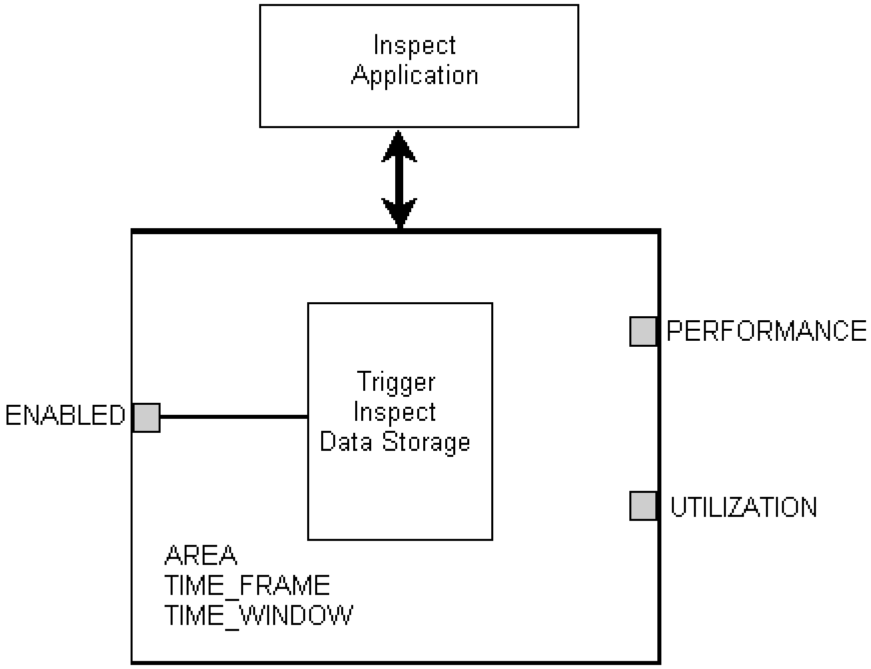
The Inspect function block does not support modes or status.



Inspect Function Block

Schematic Diagram - Inspect Function Block

The following diagram shows the internal components of the Inspect function block:



Inspect Function Block Schematic Diagram

Block Execution - Inspect Function Block

The Inspect block has no algorithm. All processing is performed in the Inspect server, which reports its results to the Inspect block.

The Inspect block has two external input parameters, ENABLED and PROCESS_IN. ENABLED turns the storing of performance data for the block on and off. Turning the storage off removes the data from the overall performance calculations. You can configure the module that contains the Inspect block to have the discrete output of another block wired to the ENABLED parameter of the Inspect block. In this way, you can programmatically control whether the data for the block is included in the overall performance calculations. When the ENABLED flag is set to 0, the Inspect server stops processing the Inspect function block.

PROCESS_IN is the input value the Inspect block averages to produce the PROCESS value. PROCESS_IN can be tied to a CALC block, a process value, or an OPC value. It is possible to have more than one Inspect block in a part of the control system (for example, if you want information for both Current Hour and Current Shift to be available). However, do not define PROCESS_IN in more than one block in the same part of the control system. If more than one Inspect block has PROCESS_IN defined the application displays only one PROCESS value and there is no way to determine which is displayed.

Parameters - Inspect Function Block

The following table lists the system parameters for the Inspect function block:

Inspect Function Block System Parameters

Parameter	Unit	Description
ADVISORY	None	The number of devices being monitored that have the Advisory flag set.
AREA	None	**Note** Leave this parameter blank. This parameter exists for compatibility with configurations created with earlier versions of the software. To report data, insert an Inspect block in a module. If the block's SYSTEM parameter is set to TRUE, the block will report statistics for the entire control system, regardless of where it is located. Otherwise, the block reports statistics for the part of the control system where the block is located.
COMMFAIL	None	The number of devices being monitored that have the Communications Failure flag set.
DEVICES	None	The total number of devices being monitored.
ENABLED	None	Turns on the storing of performance data and utilization data for the block.
FAILED	None	The number of devices being monitored that have the Failed flag set.
MAINT	None	The number of devices being monitored that have the Maintenance flag set.
PERFORMANCE	Percent	An indicator of overall performance calculated as 100 - Average Variability Index for Control blocks.
PROCESS	The units set in PROCESS_SCALE	The averaged value of PROCESS_IN over the configured TIME_FRAME and TIME_WINDOW.
PROCESS_IN	The units set in PROCESS_SCALE	The input value the Inspect block averages to produce the PROCESS value. PROCESS_IN can be tied to a CALC block, a process value, or an OPC value.

Parameter	Unit	Description
PROCESS_LABEL	None	The label for the process value of the inspect block. The label, value, and units appear in the Overview page.
PROCESS_SCALE	None	The scale and units for the PROCESS value. The label, value, and units appear in the Inspect Overview page.
SYSTEM	None	Set to TRUE on ***only*** one Inspect block in the configuration to have Inspect monitor the entire system. Set to FALSE to have the Inspect block monitor only the current module.
TIME_FRAME	None	Now, Current, or Previous
TIME_WINDOW	None	Hour, Shift, or Day
UTILIZATION	Percent	Average Percent Time that Control blocks were in their configured Normal Mode.

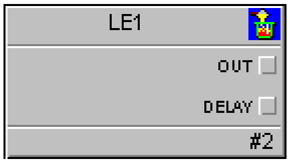
Lab Entry (LE) Function Block

Use the LE function block to provide for operator input of offline lab analysis in a DeltaV system (for example, in a Neural Network module). You define all parameters in the LE block using Control Studio.

To provide an interface that operators use to enter lab analysis results, create DeltaV operator pictures that contain an LE dynamo for each LE function block. Operators use the LE dynamo to enter the lab analysis value and the time the grab sample was taken. The block computes the delay between the time the grab sample was taken and the time the operator entered the analysis results. If both the analysis value and the calculated delay are within their maximum and minimum ranges, the dynamo writes the sample value to the OUT parameter of the LE block and writes the calculated delay to the DELAY parameter of the LE block. If either of these values is outside its valid range, a dialog notifies the operator that the entry was not accepted. However, the maximum delay that affects the Neural Netword block's automatic correction mechanism is established by the Time to Steady State configured in the DeltaV Neural application.

The LE block uses the OUT status limit attribute to indicate when an operator has entered a new lab sample. After an operator enters a new value, the block changes the OUT limit attribute from Constant to Not Limited for two block executions. Downstream blocks can use this status to detect when the OUT parameter contains the results of a new analysis.

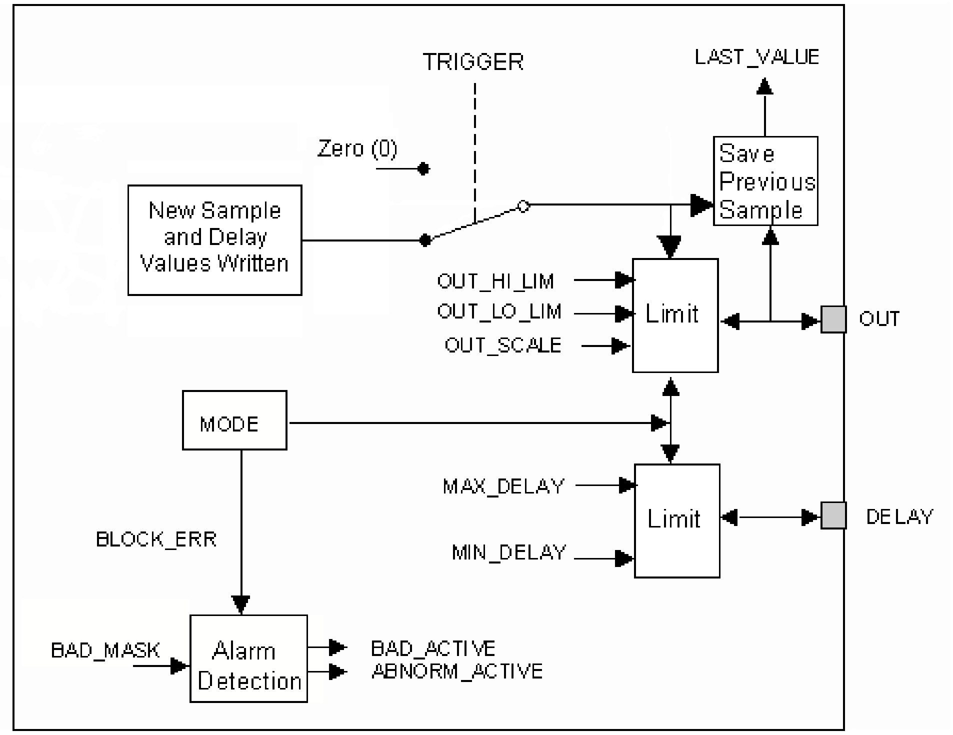
The LE block supports mode and status handling.



Lab Entry Function Block

Schematic Diagram - Lab Entry Function Block

The following diagram shows the internal components of the Lab Entry function block:



Lab Entry Function Block Schematic Diagram

Block Execution - Lab Entry Function Block

The LE dynamo writes the lab analysis and processing delay values to the LE block OUT and DELAY parameters. Operators enter the time the sample was taken in the respective fields in the dynamo. The DELAY value then is the difference (in seconds) between the sample time and the time at which its value was entered. If the LE block is in Man mode, when a new sample value is written the previous sample value is transferred to the LAST_VALUE parameter. In addition, the OUT status is changed from Good Constant to Good Not Limited for the next two (2) executions of the function block.

When the LE block mode is changed to Out of Service, the block changes the status of OUT and DELAY to BAD. In addition, the Out of Service indication in BLOCK_ERR is set. Your configuration of BAD_MASK determines whether this condition causes BAD_ACTIVE and ABNORMAL_ACTIVE to be set.

Modes - Lab Entry Function Block

The LE function block supports two modes:

Out of Service (OOS)

Manual (Man)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Lab Entry Function Block

The status of OUT and DELAY parameters is normally Good Constant when the block mode is Man. However, when a new analysis value is entered, the status of OUT changes to Good Not Limited for two executions of the block.

If the block mode is Out of Service, then OUT and DELAY have a Bad status.

Parameters - Lab Entry Function Block

The following table lists the system parameters for the Lab Entry function block:

Lab Entry Function Block System Parameters

Parameter	Unit	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions is True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions not included in BAD_MASK is True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The actual error status of the function block.
DELAY	Seconds	The time elapsed between taking a grab sample and entering the lab analysis results in the block. This value should be between MIN_DELAY and MAX_DELAY.
LAST_VALUE	EU of OUT_SCALE	The value of the previous sample value written to OUT.
MAX_DELAY	Seconds	The maximum allowed time delay in processing a grab sample for the analysis value to be accepted. This can be a maximum of 86,400 seconds (24 hours). However, the maximum delay that affects the NN block's automatic correction mechanism is established by the Time to Steady State configured in the DeltaV Neural application.
MIN_DELAY	Seconds	The minimum allowed time delay in processing a grab sample for the analysis value to be accepted. Valid range is from 0 to MAX_DELAY.

Parameter	Unit	Description
MODE	None	Parameter used to show and set the block operating state. MODE contains the actual, target, permitted, and normal modes.
OUT	EU of OUT_SCALE	A measurement value obtained through lab analysis of a grab sample.
OUT_HI_LIM	EU of OUT_SCALE	The maximum lab entry value that may be written to OUT.
OUT_LO_LIM	EU of OUT_SCALE	The minimum lab entry value that may be written to OUT.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
TRIGGER	None	Sets the current SAMPLE and DELAY values for use by the Neural Network block's automatic correction. If enabled, these values will be used by the NN function block to modify the CORR_BIAS value. Resets after two block executions.

Model Predictive Control (MPC) Function Block

The DeltaV Model Predictive Control (MPC) function block allows interactive processes to be controlled within measurable operating constraints while accounting for process interaction and measurable disturbances.

The MPC function block is the basis of implementing multivariable control in a DeltaV system. This function block replaces standard PID control that utilizes feedforward, decoupling networks, and override for multivariable control. The user defines all parameters (manipulated, disturbance, constraint, and controlled) in the MPC function block for the desired process using Control Studio. The user can launch the DeltaV Predict application from Control Studio.

The DeltaV Predict application is used to commission the MPC function block. When a module that contains the MPC block is downloaded, all inputs and outputs are assigned to the Continuous Historian.

Note Do **not** change the default history collection settings of the MPC function block.

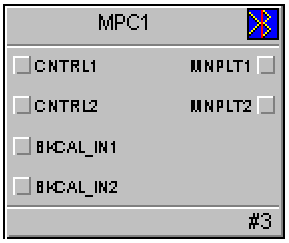
Through the Predict application, you can change the manipulated inputs to the process so that you can collect historian data that reflects the process response to these changes. Using this historian data, the Predict application can identify the process step response to changes in the manipulated or disturbance inputs to the process. Also, the control definition used by the MPC function block can be generated from the model that is identified. Refer to the DeltaV Predict topic for more information.

The operator runs the control through a DeltaV Operate screen constructed with the MPC dynamos. There are six MPC-related dynamo sets in the DeltaV Dynamo Reference Library.

Using the MPC Process Simulator function block in conjunction with the MPC block, you can create an operator training system offline to demonstrate the control that the MPC block can provide.

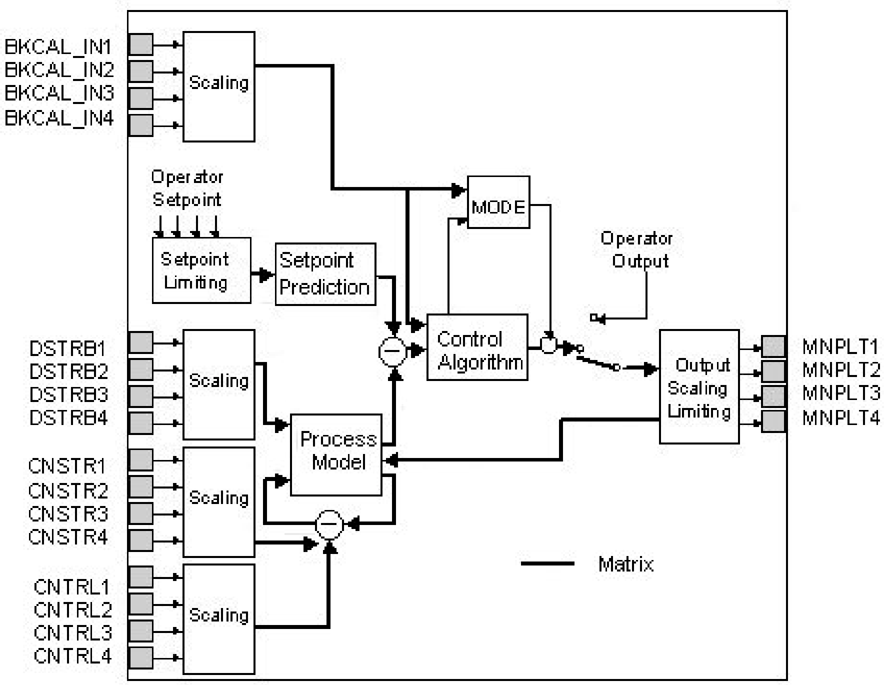
The MPC function block supports modes and status handling.

Note The MPC, AI, AO, and Scaler function blocks as well as the control blocks (PID, Ratio, Fuzzy, Bias/Gain) that are wired to the MPC block normally reside within the same module. Use the Scaler function block when there is another function block that does not support scaling (such as a Calc/Logic function block) that should be wired as an input to the MPC function block. Insert the Scaler function block between the Calc/Logic block and the MPC function block. This provides the appropriate parameter (OUT_SCALE) for the input parameters on the MPC block. When you wire an MPC function block's input or output to a function block in another module, use the MPC_INREF and MPC_OUTREF templates. These templates are located in the Advanced Control palette in Control Studio. Refer to the MPC Reference Templates topic for more information.



MPC Function Block

Schematic Diagram – MPC Function Block



MPC Function Block Schematic Diagram

Block Execution – MPC Function Block

The MPC function block defines the manipulated and disturbance inputs to a process and the controlled and constraint outputs of a process for the implementation of multivariable control strategies. This block can be used to replace conventional implementations involving decoupling, feedforward control, override control, and complex dynamics.

The target mode of the MPC function block and the status of the block's inputs determine the actual mode of the MPC block and the output status. In addition, because of the communication between the downstream function blocks and the MPC function block, the previous status of the BKCAL_IN influences the mode of the block.

The MPC algorithm executes in the following order:

1 It evaluates the actual mode based on target mode and the status of block inputs.

2 It executes the block algorithm based on the actual block mode.

3 It determines the status and value for block outputs update.

The processing done at each step of execution is described in the following sections.

Evaluate Actual Mode

The mode of the MPC function block depends on the current status of inputs, the past mode of the function block, prediction time, and the current target mode.

The manner in which the block executes depends on the actual mode of the block.

- In OOS mode, the block algorithm is not executed.
- In IMan mode, the projected response over the horizon is calculated based on the current inputs, assuming that the outputs remain at their current value. The MPC output value is calculated to match the associated BKCAL_IN inputs.
- In Man mode, the projected response over the horizon is calculated as in IMan mode. The MPC output value is calculated to match the current block output values.
- In Auto mode, the projected response is calculated and the calculated outputs for normal control generated. If the output is not limited (as indicated by its associated BKCAL_IN), the calculated output is reflected in the block output. Also, when the output is limited but the calculated output will drive the output away from the limit, the calculated output is reflected in the block output. Otherwise, the last output is maintained and the MPC algorithm re-run for the output constant.

Update Output Value and Status

The manner in which the MPC function block outputs are updated depends on the mode of the MPC block and whether a cascade connection has been established with a downstream block.

- In OOS mode, the status of the MPC outputs is set to BAD Out of Service. The block does not change the value of block outputs.
- In IMan mode, the MPC function block has been unable to establish a cascade connection with one or more of the downstream blocks because they are not in the correct mode. Until the initialization process is complete for an output, the calculate output (provide by the MPC algorithm to match the value on its associated BKCAL_IN) is written to the block output. Once the initialization process is complete for an input, the output remains at its last value with a status of GOOD Cascade Const.
- In Man mode, the output values are not written by the MPC algorithm but can be changed by the operator. The status of the outputs is GOOD Cascade Const.
- In Auto mode, the calculated output is written to the output or maintained at last value if the calculated output will drive the output further within its downstream limit. The output status should be Good Cascade NonSpecific.

MPC Function Block Execution Rate

MPC can be used in an application as long as a function block execution time of one second or slower is sufficient to meet your process control requirements. The MPC function block scan rate multiplier is automatically set based on the time to steady state to provide an algorithm execution rate of one second or slower. Since the MPC algorithm uses model and controller matrices, which are generated on a very specific time base, the DeltaV Predict application automatically configures the effective scan rate. The execution time is defined by the process response time estimation (entered as the time to steady state when you request that the process be tested for identification) divided by prediction horizon (120 samples) in seconds.

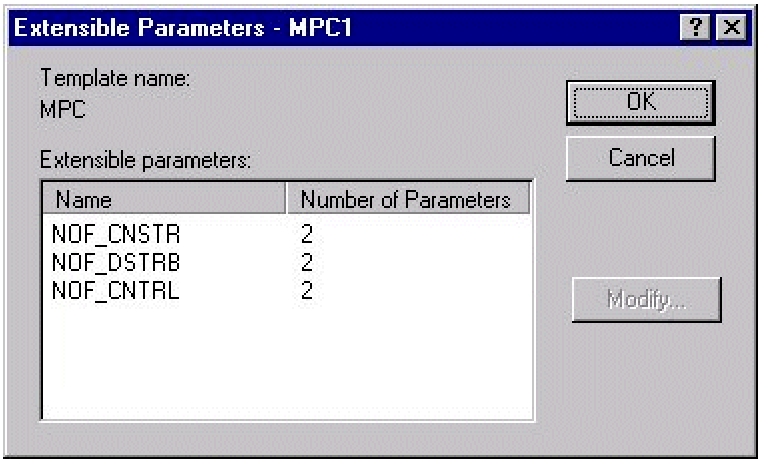
Alarming

When the MPC function block detects configuration errors or abnormal conditions, they are automatically reflected in the BLOCK_ERR parameter. By setting the BAD_MASK parameter, you can determine which condition set BAD_ACTIVE. All conditions not included in BAD_ACTIVE are reflected in the ABNORMAL_ACTIVE parameters. These xxxx_ACTIVE parameters can be used in defining module alarms.

MPC Configuration

Note Any change in the block configuration requires that you reconnect to the block from the Predict application for the change to be reflected in the block's behavior.

Since the MPC block is an extensible function block, you can select the number of controlled, constraint, and disturbance parameters that are required to meet your process control needs. The number of manipulated and back calculation input parameters are set to equal the number of controlled parameters that you specify. The default is for the MPC block to have two controlled parameters. When you right-click the MPC function block and select Extensible Parameters, the following dialog is provided to specify MPC block inputs.



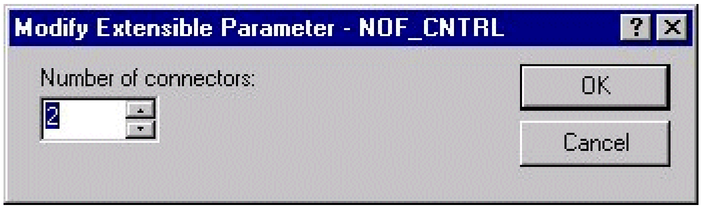
Extensible Parameters - MPC1

Template name:
MPC

Extensible parameters:

Name	Number of Parameters
NOF_CNSTR	2
NOF_DSTRB	2
NOF_CNTRL	2

When you select the number of either control, disturbance, or constraints (NOF_XXXX) and then select Modify, a dialog appears in which you can change the number of inputs. For example, to change the number of controlled parameters, select NOF_CNTRL and then click Modify. The following dialog appears, which allows you to specify the number of controlled parameters.

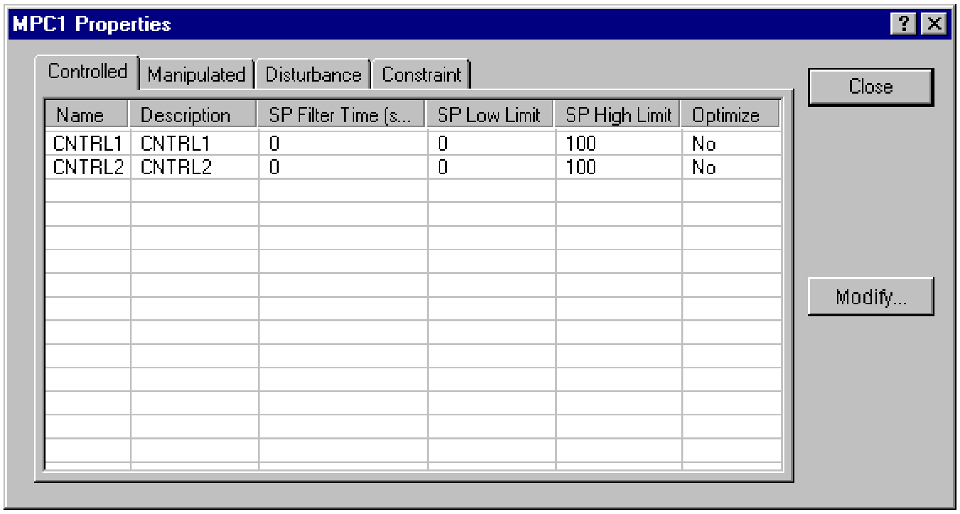


Modify Extensible Parameter - NOF_CNTRL

Number of connectors: 2

After you specify the number of inputs to meet your application needs, select OK. The dialog closes, and the MPC block resizes to reflect the specified number of inputs (and outputs).

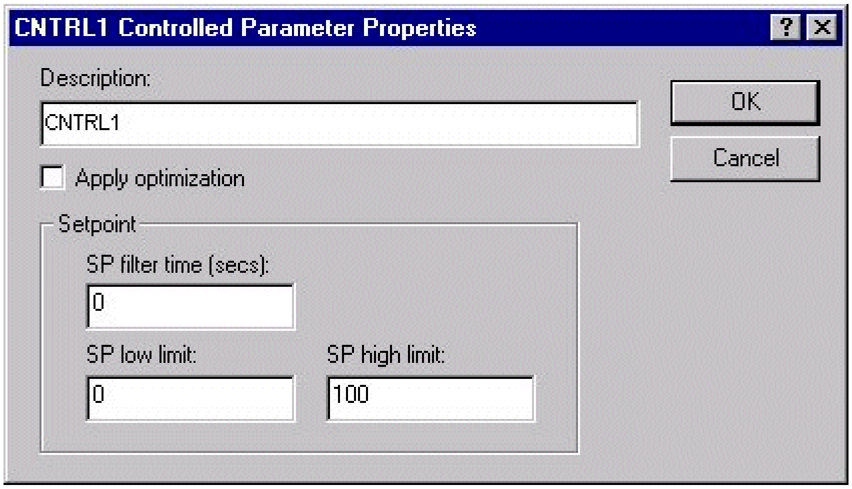
Once you define the number of inputs to the MPC block, you can configure the parameters associated with this function block. To enter configuration information, right-click the MPC function block and then select Properties…. In response, the following dialog appears, which allows you to enter configuration information.



Name	Description	SP Filter Time (s...	SP Low Limit	SP High Limit	Optimize
CNTRL1	CNTRL1	0	0	100	No
CNTRL2	CNTRL2	0	0	100	No

You can specify the configurable parameters associated with the controlled, manipulated, disturbance, and constraint parameters. Click an input or output listed in this dialog and select the Modify button. Then, you can change the configurable values associated with this parameter.

For a controlled parameter, the following configuration dialog appears.



Description: CNTRL1

□ Apply optimization

SP filter time (secs): 0

SP low limit: 0

SP high limit: 100

Provide the following configuration information:

Description – The parameter name that you want to appear in the Predict application and the MPC Operate view. (For more information, refer to the MPC Dynamos sections in the Dynamo Reference topic.) The default description name is the connector name (for example, CNTRL1). It is recommended that the parameter description contain 14 characters or fewer.

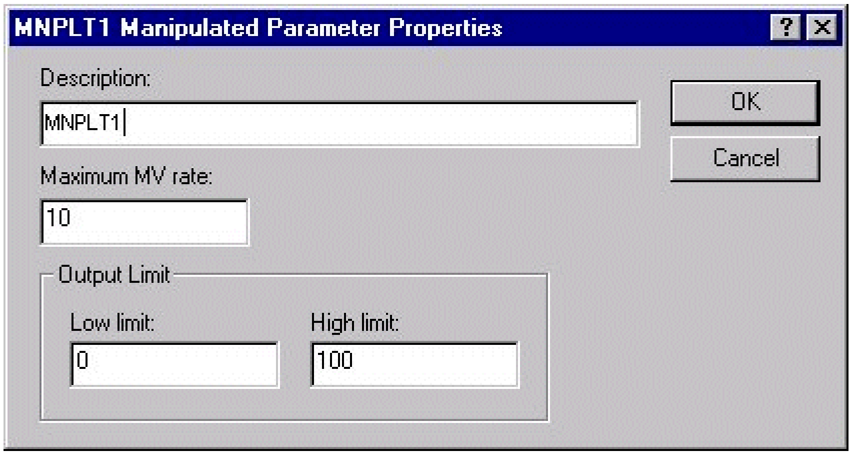
SP Filter Time – The time required for 63% of an SP change to be made (in seconds). This filter time is used to set the setpoint trajectory.

SP Low Limit – The minimum SP value (in engineering units) that an operator can enter. A setpoint entry that is lower than this limit is truncated to the limit value.

SP High Limit – The maximum SP value (in engineering units) that an operator can enter. A setpoint entry that is greater than this limit is truncated to the limit value.

Apply Optimization (check box) – Only select this feature if the manipulated parameter associated with this controlled parameter is used to set the throughput of the process. If this option is selected, the associated manipulated parameter is adjusted to maintain the controlled parameter either at setpoint or a value that maintains a manipulated input being maintained at a its limit, whichever is lower.

For a manipulated parameter, the following configuration dialog appears.



MNPLT1 Manipulated Parameter Properties

Description: MNPLT1

Maximum MV rate: 10

Output Limit

Low limit: 0

High limit: 100

Provide the following configuration information:

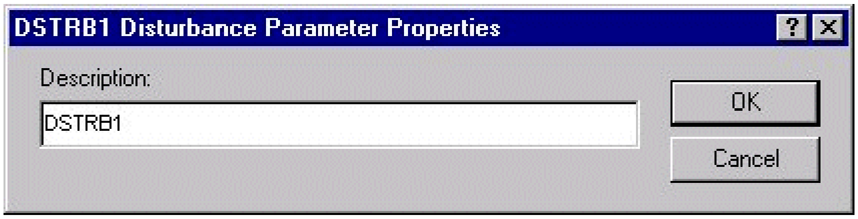
Description – The parameter name that you want to appear in the Predict application and the MPC Operate view. (For more information, refer to the MPC Dynamos sections in the Dynamo Reference topic.) The default description is the connector name (for example, CNTRL1). It is recommended that the parameter description contain 14 characters or fewer.

Maximum MV Rate – The maximum rate at which the MPC function block can change the manipulated parameter (in engineering units per second).

Low Limit – The minimum output value (in engineering units) that an operator can enter. An entry that is lower than this limit is truncated to the limit value.

High Limit – The maximum output value (in engineering units) that an operator can enter. An entry that is greater than this limit is truncated to the limit value.

For a disturbance parameter, the following configuration dialog appears.



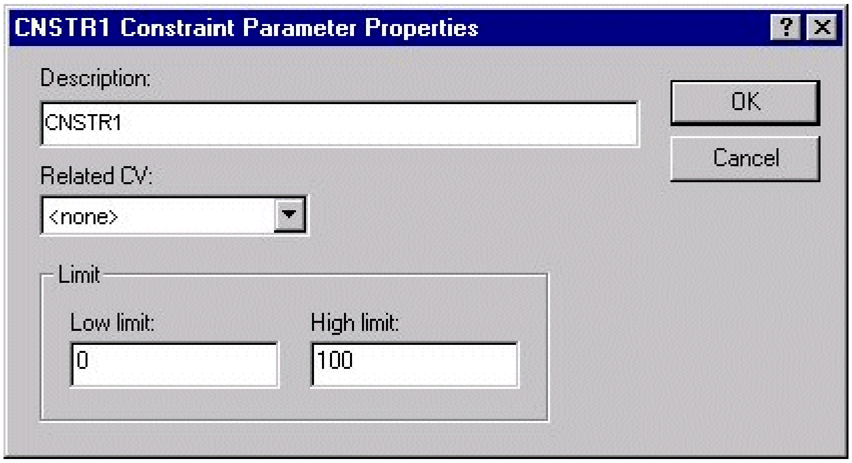
DSTRB1 Disturbance Parameter Properties

Description: DSTRB1

Provide the following configuration information:

Description – The parameter name that you want to appear in the Predict application and the MPC Operate view (For more information, refer to the MPC Dynamos sections in the Dynamo Reference topic.) The default description is the connector name (for example, CNTRL1). It is recommended that the parameter description contain 14 characters or fewer.

For a constraint parameter, the following configuration dialog appears..


CNSTR1 Constraint Parameter Properties

Description: CNSTR1

Related CV: <none>

Limit

Low limit: 0

High limit: 100

Provide the following configuration information:

Description – The parameter name that you want to appear in the Predict application and the MPC Operate view (For more information, refer to the MPC Dynamos sections in the Dynamo Reference topic.) The default description is the connector name (for example, CNTRL1). It is recommended that the parameter description contain 14 characters or fewer.

Related CV – The controlled parameter that is allowed to deviate from setpoint to prevent the constraint from violating its low or high limit.

Low Limit – The minimum constraint target (in engineering units) that an operator can enter. An entry that is lower than this limit is truncated to the limit value.

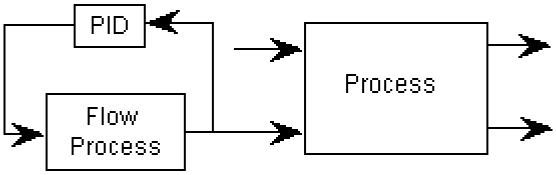
High Limit – The maximum constraint target (in engineering units) that an operator can enter. An entry that is greater than this limit is truncated to the limit value.

Wiring to an MPC Function Block

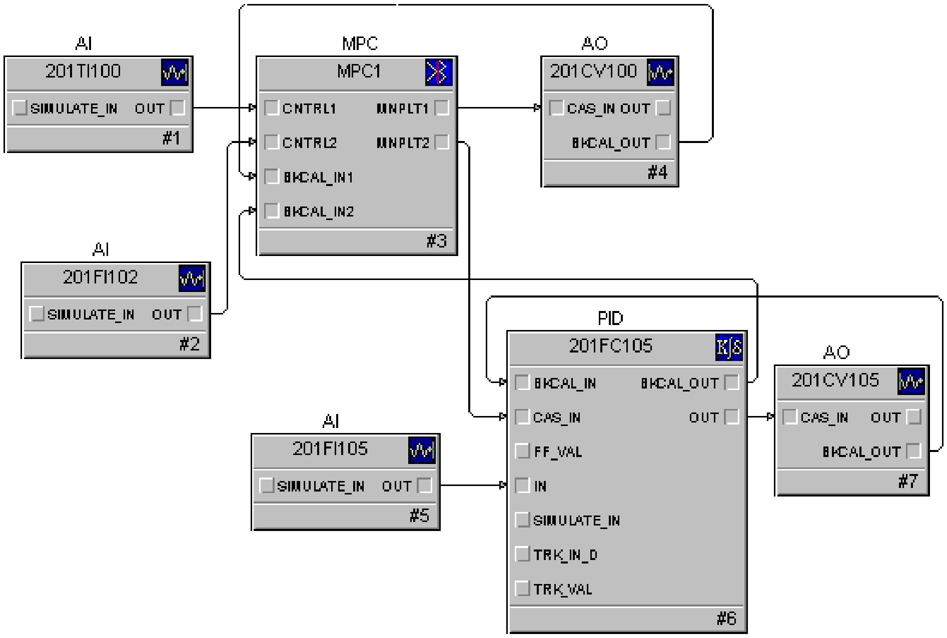
The MPC function block may be wired to input and output blocks that are assigned to traditional or Hart transmitters and valves. It can also be used with blocks that are assigned to fieldbus devices. All function blocks connected to the MPC function block must contain a scaling parameter. During MPC execution, this scaling information is automatically read by the MPC block and used in input-output scaling within the block. If an MPC block's inputs are connected to a block that does not contain a scaling parameter (for example, a math block) a Scaler function block is required. The Scaler function block is directly wired to both the MPC and upstream function blocks so that the MPC block can access the scaling information.

All function blocks that are wired to the MPC block are expected to be within the same module. If you are connecting an input or output to a block in another module, use the MPC_INREF or MPC_OUTREF templates. Refer to the MPC Reference Templates topic for more information.

The MNPLT outputs must be wired directly to the CAS_IN, RCAS_IN, or ROUT_IN inputs of the downstream function blocks. If the MPC block is being used to replace a PID block, it is typically wired to an AO block. However, for many other applications, the output of the MPC function block is wired as the cascade input of a PID, Fuzzy Logic or Ratio function block that provides the process input. An example of such wiring is illustrated in the following figure.

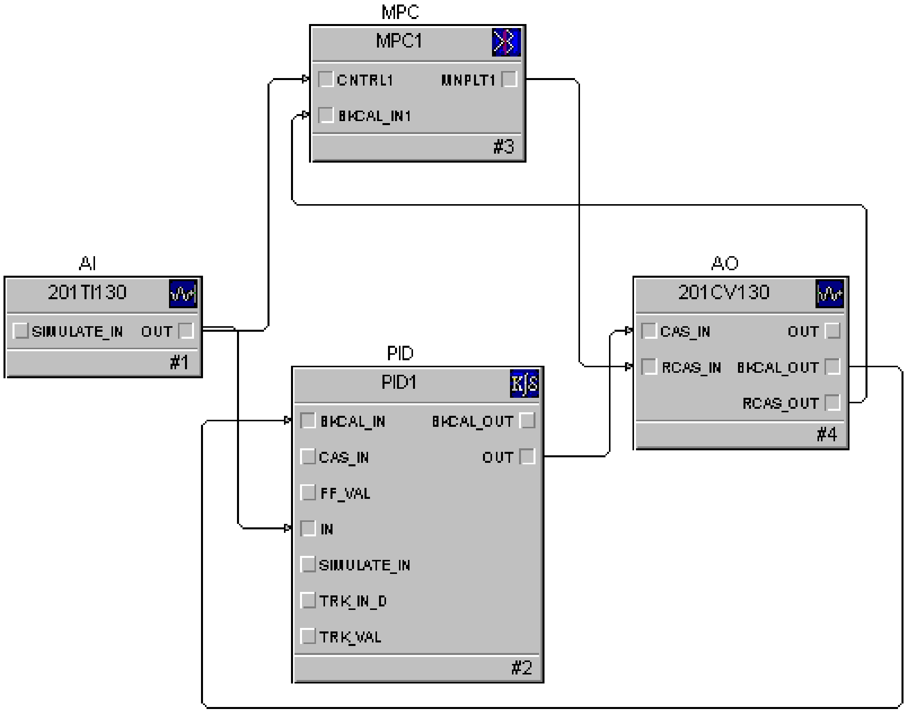


In such cases, the downstream process setpoint is treated by the MPC function block as the manipulated process input. As such, the MPC function block's manipulated output is wired to the CAS_IN of this function block, as shown in the following figure.



Like other DeltaV control blocks, you must wire the BKCAL_OUT (or ROUT_OUT or RCAS_OUT if the manipulated output is wired to ROUT_IN or RCAS_IN) of the downstream block to the BKCAL_IN inputs of the MPC function block.

In some cases, the MPC function block replaces an existing control strategy. You can layer the MPC function block on the existing strategy. Doing so allows the MPC control to be implemented with the old strategy as a backup. You can achieve this by exposing the AO or control block's RCAS_IN and RCAS_OUT (or ROUT_IN and ROUT_OUT) that the existing strategy is manipulating. Also, you can wire the MPC function block to these parameters in much the same way that you can wire to CAS_IN and BKCAL_OUT. An example of this wiring is shown in the following figure.



Modes – MPC Function Block

The MPC function block supports the following modes:

Out of Service (OOS) – Set by changing the target mode to OOS.

Initialization Manual (IMan) – Indicates that the path to the process is broken. A common cause is that a downstream block is not in the correct mode to utilize the output of the MPC block.

Local Override (LO) - Indicates that testing by the Predict application is in progress

Manual (Man) – Operator has full control of the MPC block outputs.

Automatic (Auto) – Outputs are automatically calculated by the MPC block.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling – MPC Function Block

If the status of a controlled, constraint, or disturbance input to the MPC block is Bad, the actual mode automatically goes to Man. UNCERTAIN status on inputs to the MPC block are treated as Good in the mode evaluation..

The BKCAL_IN input statuses are used to handshake with the downstream block for bumpless transfer and to detect that the downstream blocks are or are not in the correct mode in response to changes made by the MPC function block. The MPC function block's actual mode remains in IMan until all downstream blocks are in the correct mode and the handshake for bumpless transfer is complete.

If the mode of the MPC function block is changed to Out of Service, the status of the block outputs change to BAD Out of Service.

The MPC function block remains in IMan until all blocks downstream are in the correct mode (Cas or RCas). The actual mode of the MPC block goes to LO when Predict application is active.

Parameters – MPC Function Block

The following table lists the system parameters for the MPC function block:

MPC Function Block Parameters

Parameter	Unit	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active), an error condition (at the module level) not selected in MERROR_MASK is True (Active), or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active), an error condition (at the module level) selected in MERROR_MASK is True (Active), or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions is True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions not included in BAD_MASK is True, ABNORM_ACTIVE becomes True.
BKCAL_IN [1…4]	EU of MNPLT	The analog input value and status from another block's BKCAL_OUT output that is used by an upstream block for bumpless transfer. The number of BKCAL_IN parameters corresponds to the number of MNPLT parameters.
BLOCK_ERR	None	The actual error status of the function block.
CNSTR [1…4]	The output units of the connected block	Constrained parameter (input to MPC function block). Is a range within which the output must fall.
CNTRL [1…4]	The output units of the connected block	Controlled parameter (input to MPC function block). Is maintained at a specific value (setpoint) by adjusting the function block's manipulated output.
DSTRB [1…4]	The output units of the connected block	Disturbance inputs (to the MPC function block). The impact of disturbance inputs on controlled and constraint parameters is reduced substantially through the MPC block's adjusting manipulated inputs to compensate for changes in the disturbance inputs.

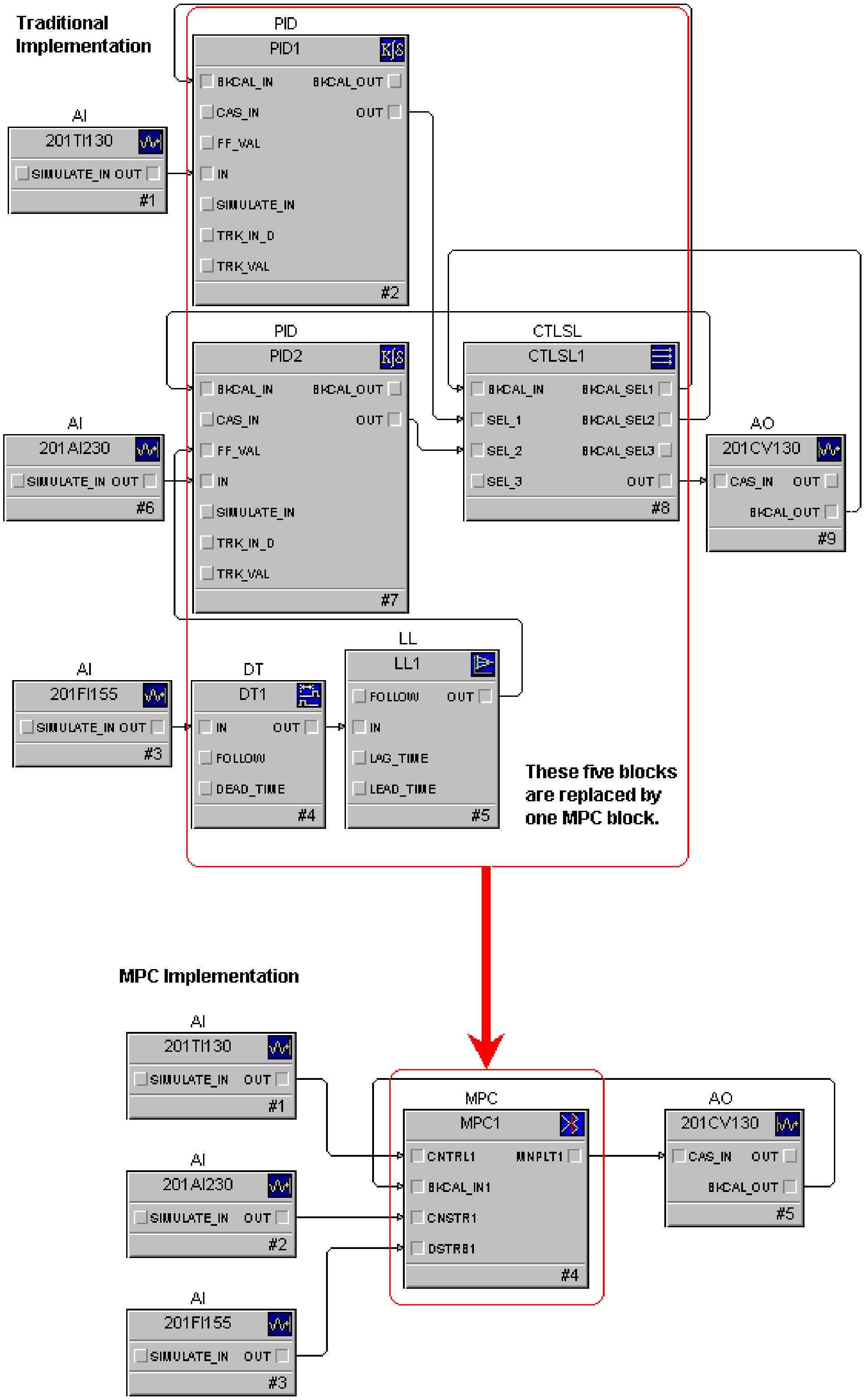
Parameter	Unit	Description
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 if the following conditions are True: - The Write To Inspect Alarm context menu item was selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition will only exist for Variability if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
MNPLT [1…4]	The input units of the connected block	Manipulated output (from the MPC function block). A variable that is adjusted either automatically by the MPC block or manually by an operator. The MPC algorithm acts on the manipulated value to maintain the constraints within limits and the controlled parameter values at setpoint.
MODE	None	Parameter used to show and set the block operating state. MODE contains the actual, target, permitted, and normal modes.
SP [1…4]	EU of CNTRL	The operator’s setpoint values. The number of SP parameters corresponds to the number of CNTRL parameters.
SP_WRK [1…4]	EU of CNTRL	The working setpoint of the block. The filtered value of the operator setpoint values used in determining control action.
STDEV	Percent of scale	The largest standard deviation calculation for the control parameters over the standard deviation time. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_CAP	Percent of scale	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.

Application Examples - MPC Function Block

The MPC function block provided by DeltaV Predict may be used to address multivariable control requirements You can use the Predict application to create process models and control definitions from process data. The automated test feature of the Predict application allows the step response of controlled or constraint outputs to be determined based on changes in the manipulated inputs. This capability can meet multivariable control requirements in all process industries. In many cases, the MPC function block can replace control techniques that have traditionally been used to address the control requirements of multivariable processes.

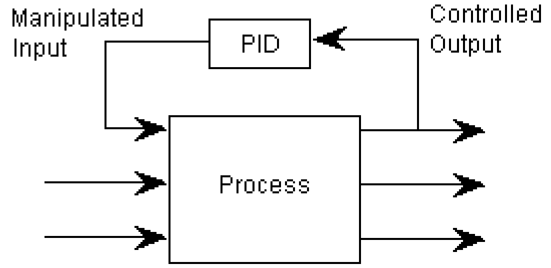
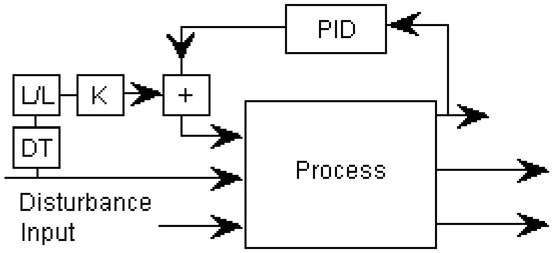
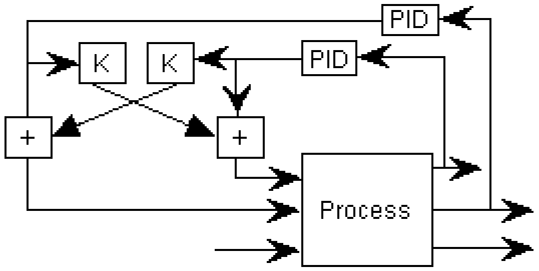
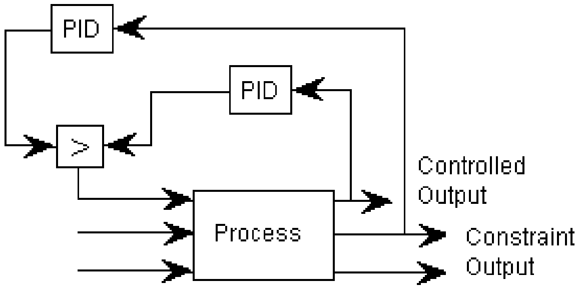
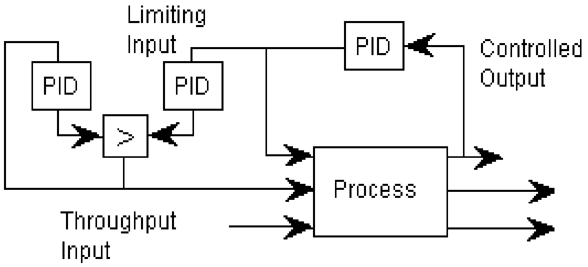
Multivariable control is defined as control that uses multiple process measurements to produce one or more outputs. The techniques available to an engineer for implementing multivariable control have been limited to the conventional capability available in most commercial process control systems. The most commonly used techniques are illustrated in the table entitled "Traditional Multivariable Control Techniques." You can combine such techniques to address the control requirements of a multivariable process. However, as the number of interacting control loops, operating constraints, and load disturbances used in the control increases, the complexity of the control system is often beyond that which can be maintained by the typical plant engineer or instrument technician. You can use a single MPC function block to address the complete requirements of many multivariable processes.

The implementation and commissioning of the MPC function block are far simpler and faster than traditional techniques, as is shown in the following figure. The difference is even more dramatic for applications involving a larger number of controlled, constraint, or disturbance parameters or the optimization of throughput.



Comparison of Traditional and MPC Usage for Application with One Control, Disturbance, and Constraint Parameter

Traditional Multivariable Control Techniques

Feedback Control - Regulation of a process input to maintain the process at a target operating setpoint. The common means of feedback control is the PID algorithm. Tuning of PID is based on the dynamic response of the process to a change in the manipulated input.	
Feedforward Control - Compensation for measured process disturbances by adjustment of process input. In general, dynamic compensation using a deadtime and lead/lag unit is required. Setup of dynamic compensation is based on the dynamic response of the process to changes in the disturbance input.	
Decoupling Network - Accounting for process interaction in the adjustment of process inputs. When manipulated inputs impact the process in the same time frame, gain compensate based on the relative gain associated with each controlled parameter may be utilized. If the dynamics are different, a lead/lag and deadtime unit are required.	
Override Control - Maintenance of feedback control within operating constraints. When the constraint exceeds its setpoint, the output of the override PID is selected for control. Tuning of the constraint PID is based on the dynamic response of the constraint parameter to a change in the associated manipulated input.	 PID PID > Controlled Output Process Constraint Output
Throughput Optimization - Throughput is maintained either at its setpoint or at a limit established by a process input. When the input limit is reached, the throughput is reduced to maintain the process at the input limit. This way, the maximum throughput may be maintained.	 Limiting Input PID Controlled Output PID PID > Process Throughput Input

Applications that may benefit from MPC control can be identified based on your process knowledge. The following guidelines may be used to determine when MPC should be considered.

1 Single or multiple loop control that is characterized by either long delay or inverse process response. Since the control action taken by an MPC function block is based on a process response model, you can achieve better control than would be possible with PID feedback control or deadtime compensation techniques, such as the Smith Predictor. The response model used by MPC is determined by the Predict application during commissioning.

2 The interaction of two or more control loops impact process operation (that is, a change in the output of one control loop impacts the control parameter of the other control loops in a significant manner). Such interaction is accounted for when control action is taken by the MPC function block. Therefore, control may be improved since any correction that is taken to adjust one control parameter only impacts that parameter.

3 One or more disturbances to any controlled parameters are measured. By including these measurements in the control, the MPC function block compensates for the impact of disturbances. As such, changes in the disturbance input have little or no impact on the controlled parameters.

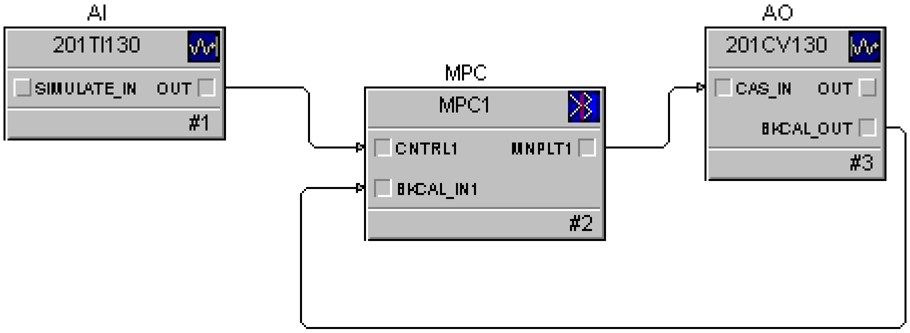
4 One or more measurable constraints must be observed in control of the process. The MPC function block constantly calculates the impact of a disturbance or control action on constraint parameters. If the future value of a constraint output violates its limit, MPC takes the appropriate action to prevent the constraint limit from being violated.

5 Production is limited by one or more inputs to a process. Control of the throughput may be included in the MPC along with the control associated with the inputs that limit production. Throughput is adjusted to maintain the process at its input limit and achieve maximum production.

The execution rate of the MPC function block is limited to one second or slower. Therefore, MPC should be applied to processes where control requirements may be satisfied at these execution speeds.

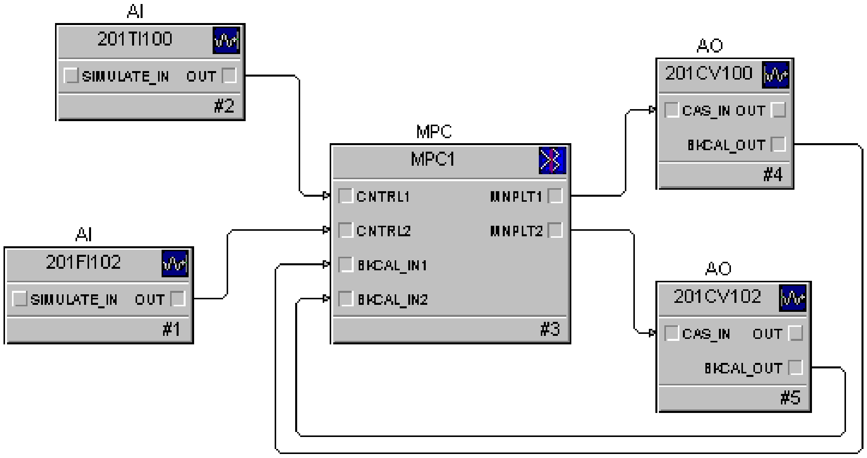
Difficult Process Dynamics

When the process response to a change in the manipulate input either is dominated by delay or exhibits an inverse response, regulation provided by PID control may be too sluggish to meet your control requirements. In such cases, you can use the MPC function block for this control application. Since the control provided by MPC is based on an accurate model of the process, you may achieve tighter control than can be achieved with PID. An example implementation of MPC for single loop control is shown in the following figure.



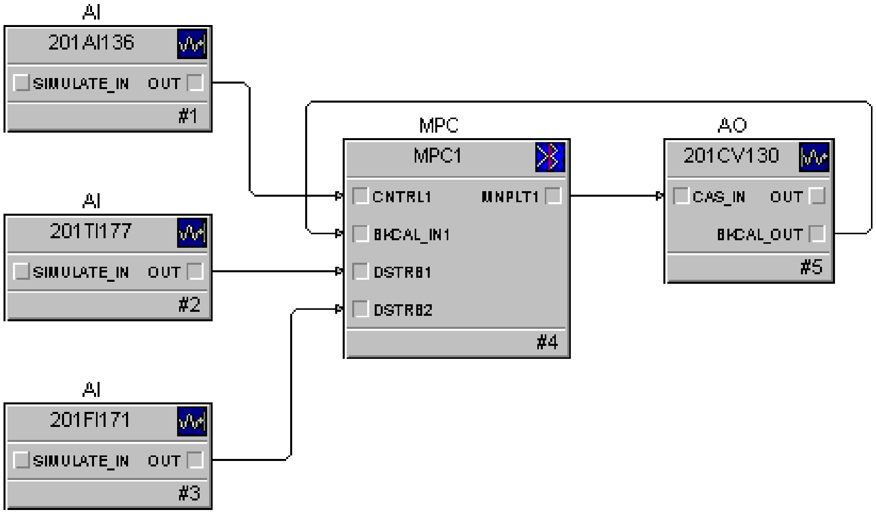
Interacting Control Loops

The model that is developed for MPC control captures the impact of a change in manipulated and disturbance inputs on all of the process outputs. Therefore, the MPC function block can correct for a deviation in one controlled parameter without impacting other controlled parameters associated with the block. Such control can be implemented and commissioned more quickly and easily than by using traditional techniques, such as decoupling networks, to compensate for loop interaction. An example of the control of two interacting control loops using MPC is shown in the following figure.



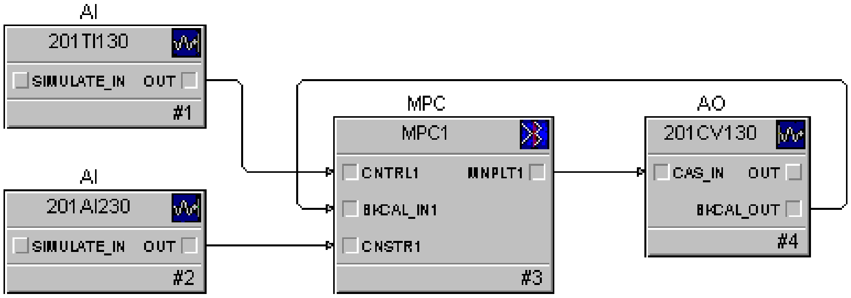
Measured Disturbances

The dynamics associated with a measured process disturbance are identified by the MPC process model. This model is the basis for MPC control. It allows the control to compensate for disturbances. No additional dynamic compensation elements are required to compensate for process disturbances (as with traditional techniques). An example of a single control loop with two measured disturbances is shown in the following figure.



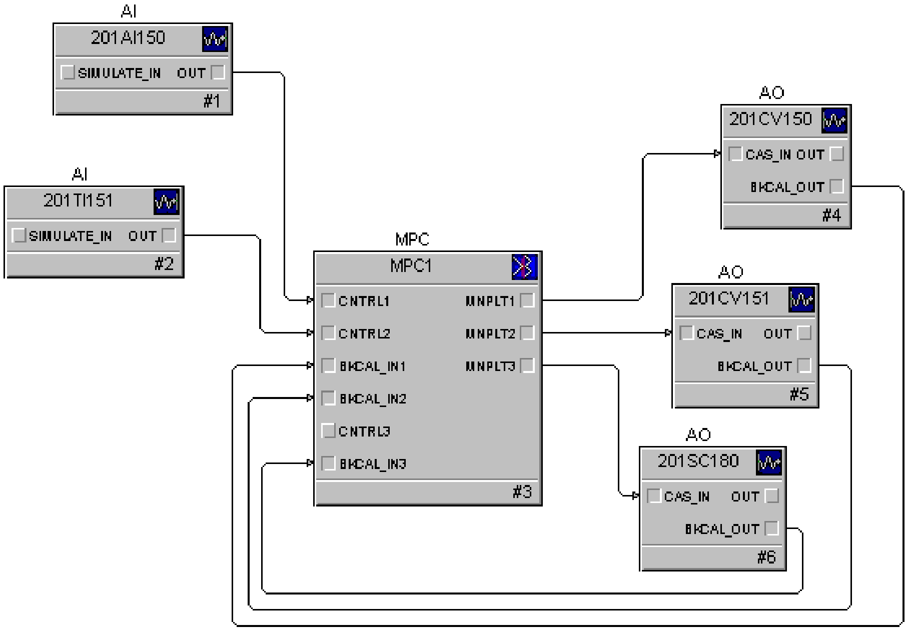
Measured Constraint

Constraint handling is an integral part of the MPC function block. The impact of disturbances and changes in manipulated inputs on the constraint may be predicted; such impact is based on the predictions made by MPC. When the value of the constraint is predicted to exceed the constraint limit defined by its setpoint, the internal target of the associated controlled parameter is reduced. This occurs so that the resulting changes in manipulated inputs prevent the constraint from exceeding its limit. As part of the constraint configuration, you must identify the associated controlled parameter and indicate whether the constraint setpoint represents a minimum or maximum limit. An example of a single control loop with one constraint is shown in the following figure.



Process Input Limits Production

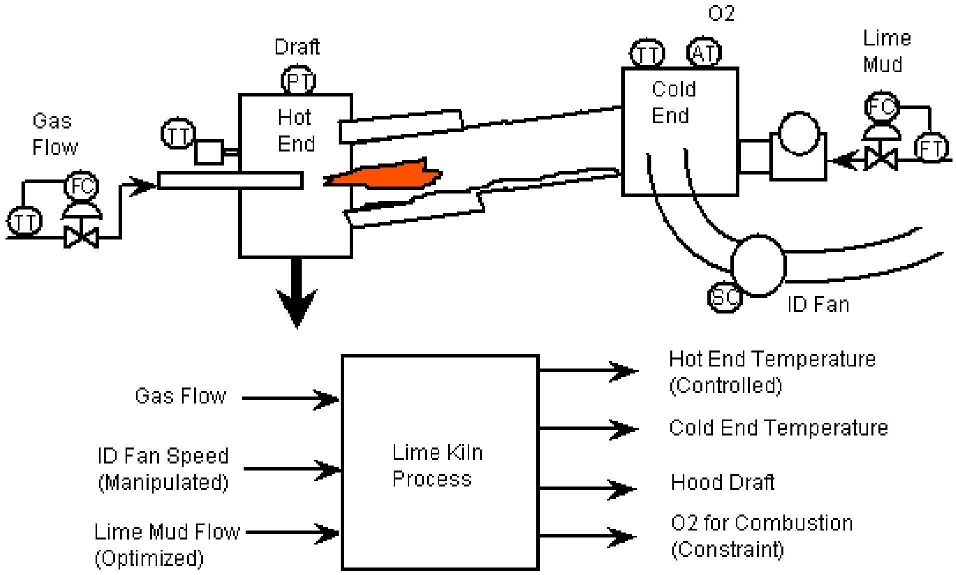
One manipulated input defined for the MPC function block may be identified as an optimizing input (that is, an input that is to be maintained either at target or at a value that keeps one of the other manipulated inputs at a defined maximum or minimum limit). When a manipulated input is selected for optimization, you must configure a minimum or maximum limit for the other manipulated inputs. Also, no connection is required for the control input associated with the manipulated output used to set throughput. When the throughput setpoint is changed, the manipulated output for throughput adjustment will be changed until the setpoint is reached. If another manipulated parameter reaches or exceeds its configured high or low limit on a change in either throughput target or during normal operations, the throughput will be adjusted to maintain the manipulated output at its limit. An example of an MPC function block being used to control two parameters as well as optimized throughput is shown in the following figure.



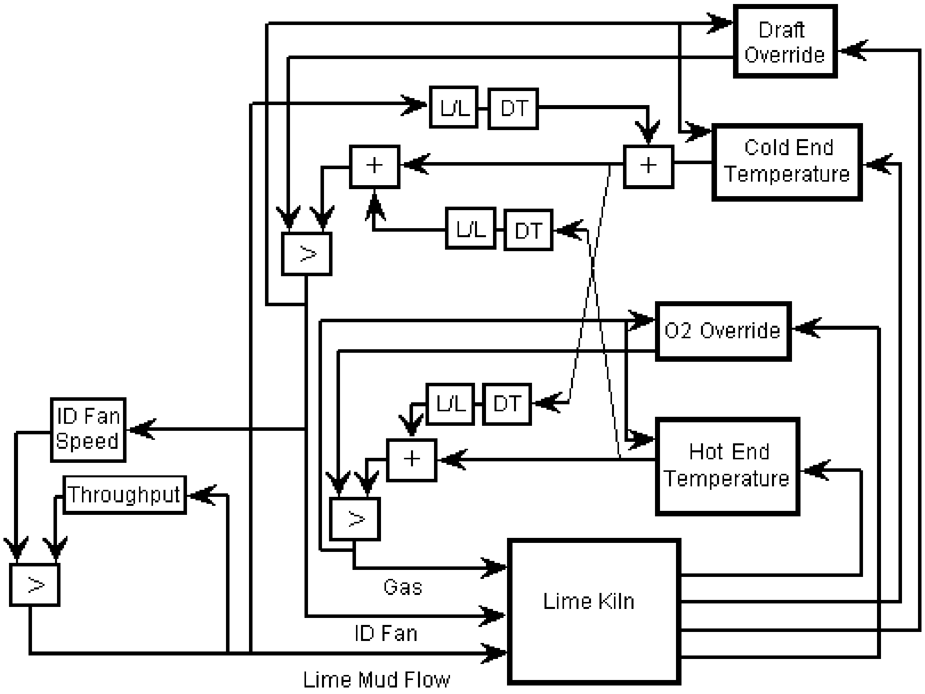
Large Multivariable Applications

The design of many processes requires that the control strategy address multiple controlled, constraint, and disturbance outputs through the adjustment of multiple manipulated process inputs. The MPC function block can control processes of size 8x8 (for example, four controlled, four constraint, four disturbance, and four manipulated parameters maximum). The lime kiln example in this topic demonstrates how the techniques used in previous examples may be combined to meet the control requirement of a more complex process.

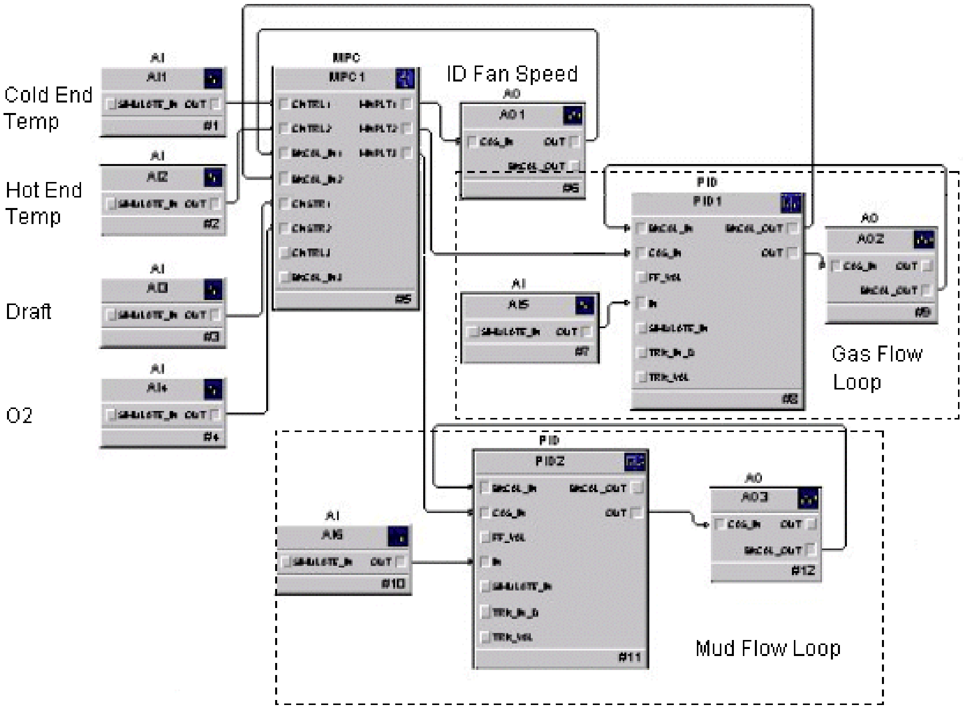
The following figure illustrates a typical lime kiln process and instrumentation for the recausticizing area of a pulp mill. This process is 3x4 in size, where gas and ID fan speed are manipulated to control the material and exit gas temperature at the hot and cold end of the kiln. Constraints on gas and ID fan speed adjustment are set by limits on O2 and hood draft, respectively. The lime mud flow throughput is maximized within the constraints of the ID fan speed limits.



A traditional control strategy for the lime kiln is shown in the following figure. For simplicity, the PID controller associated with the mud flow and gas flow are not shown but are required in the final implementation. The kiln control requires a total of six PID controllers (plus the gas and lime mud flow PID controllers), three control selectors, and nine math and dynamic elements. A decoupling network addresses the interaction between the hot and cold end temperature control. Constraints on O2 and draft are provided as overrides on gas flow and ID fan speed adjustment. Throughput is maximized based on the ID fan being the limiting factor on production.



The equivalent control is provided by one MPC function block. The complete control strategy (including the lime and mud flow control) using the MPC function block is shown in the following figure.



Lime Kiln Control Using MPC

MPC Reference Templates

The MPC function block can be connected to inputs and outputs from blocks in other modules. These special external input and output references in the MPC module are designed to provide the PV_SCALE value for inputs and the PV_SCALE and MODE values for outputs.

To reference an MPC block input or output to a block in another module, you must use the MPC_INREF template for input references and the MPC_OUTREF template for output references. Both templates are located in the Advanced Control palette in Control Studio.

MPC_INREF Template

The MPC Input Reference template is used to map MPC relevant data from a specified upstream function block into the current module. The following rules apply.

- The specified upstream function block must be an AI, a PID, an FLC, or an SCLR function block.
- The upstream function block path is entered into the PATH parameter in the //<MODULE>/<BLOCK> (upper case) format.

MPC_INREF Quick Config Parameters

Quick Config Parameters for MPC_INREF

Parameter	Description
PATH	The path of the upstream function block. The path must be entered in the following format: //<MODULE>/<BLOCK> (upper case)

MPC_OUTREF Template

The MPC Output Reference template is used to map MPC relevant data from a specified downstream function block into the current module. The following rules apply.

- The specified downstream function block must be an AO, a PID, or an FLC function block.
- The downstream function block path is entered in the PATH parameter in the //<MODULE>/<BLOCK> (upper case) format.

The MPC Output Reference composite template supports standard alarming.

MPC_OUTREF Quick Config Parameters

Quick Config Parameters for MPC_OUTREF

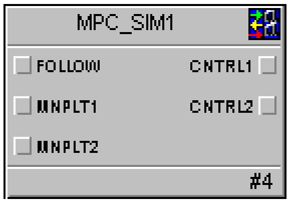
Parameter	Description
PATH	The path of the downstream function block. The path must be entered in the following format: //<MODULE>/<BLOCK> (upper case)

Model Predictive Control Process Simulator Function Block

The MPC Process Simulator function block simulates the actual process for use with the MPC function block for operator training. By including the MPC Process Simulator block in a module that contains an MPC module, you can train operators on your process without using a live process.

The MPC Process Simulator block is designed to be used with the MPC function block. This block can be used to simulate the process in a module that contains a MPC control block. The MPC simulation block is wired to the I/O blocks associated with the MPC control block. Based on the disturbance and manipulated inputs to the process, the MPC process simulation block calculates the dynamic response in the controlled and constraint outputs of the process.

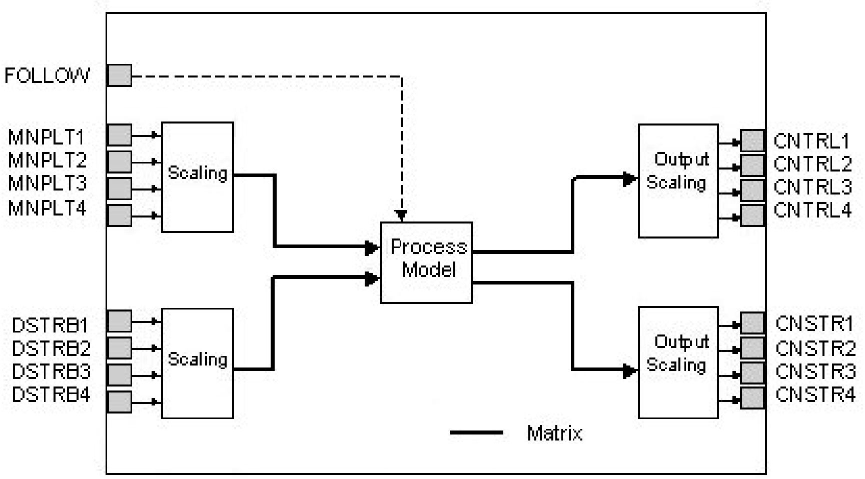
The MPC Process Simulator block allows you to simulate the dynamic response of a process. This simulation utilizes the step responses that were identified by DeltaV Predict as part of the development of the control definition for an MPC block.



MPC Process Simulator Block

- MNPLTx is the Manipulated Variable input of the process.
- DSTRBx is the Disturbance Variable input of the process.
- CNTRLx is the Controlled Variable output of the process.
- CNSTRx is the Constraint Variable output of the process.
- FOLLOW is provided to allow the process dynamics to be initialized.

Schematic Diagram - MPC Process Simulator Function Block



MPC Process Simulator Function Block Schematic Diagram

Block Execution - MPC Process Simulator Function Block

The simulate block inputs and output should be defined to match those of the MPC block in the module. Once this is done, a selected model file created using the Predict application may be assigned to an MPC Process Simulator block. To make this assignment, right-click the MPC Process Simulator function block and select Predict. From the Predict application, click File | Assign Simulation Model to select a model for use in the simulation. The MPC Process Simulator block calculates the process output based on the disturbance and manipulated input values provided by the MPC block as well as the assigned process model.

Initialization - MPC Process Simulator Function Block

If FOLLOW is enabled, the outputs of the MPC Process Simulator function block are initialized to their steady-state value based on the input values.

Parameters - MPC Process Simulator Function Block

The following table lists the system parameters for the MPC Process Simulator function block:

MPC Process Simulator Function Block Parameters

Parameter	Unit	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active), an error condition (at the module level) not selected in MERROR_MASK is True (Active), or a module status not selected in MSTATUS_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active), an error condition (at the module level) selected in MERROR_MASK is True (Active), or a module status selected in MSTATUS_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions is True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions not included in BAD_MASK is True, ABNORM_ACTIVE becomes True.
CNSTR [1…4]	The input units of the connected block.	Constrained parameter (inputs to the MPC function block). Process output to be maintained within a specified range of operation.

Parameter	Unit	Description
CNTRL [1…4]	The output units of the connected block.	Controlled parameter (inputs to the MPC function block). Process output that is to be maintained at a specific target value.
DSTRB [1…4]	The output units of the connected block.	Disturbance parameter. Process input that is determined by upstream process condition and impacts controlled and constraint parameters.
FOLLOW	None	Allows the output to track the input or the input times gain (True = Active; False = Inactive). When enabled, this parameter is used to force the MPC Process Simulator function block to a steady state condition.
MNPLT [1…4]	The output units of the connected block.	Manipulated parameter (output from the MPC function block). Process inputs that are adjusted by the MPC block in automatic operation or by an operator in manual mode.

Application Example - MPC Process Simulator Function Block

The MPC Process Simulator function block may be used in conjunction with the MPC function block to implement an operator training environment in an offline system (for example, workstation running DeltaV Simulate). The steps required to create a dynamic simulation of the control are as follows:

1 You can import the module that contains the MPC function block of interest into the training system along with the associated MPC models.

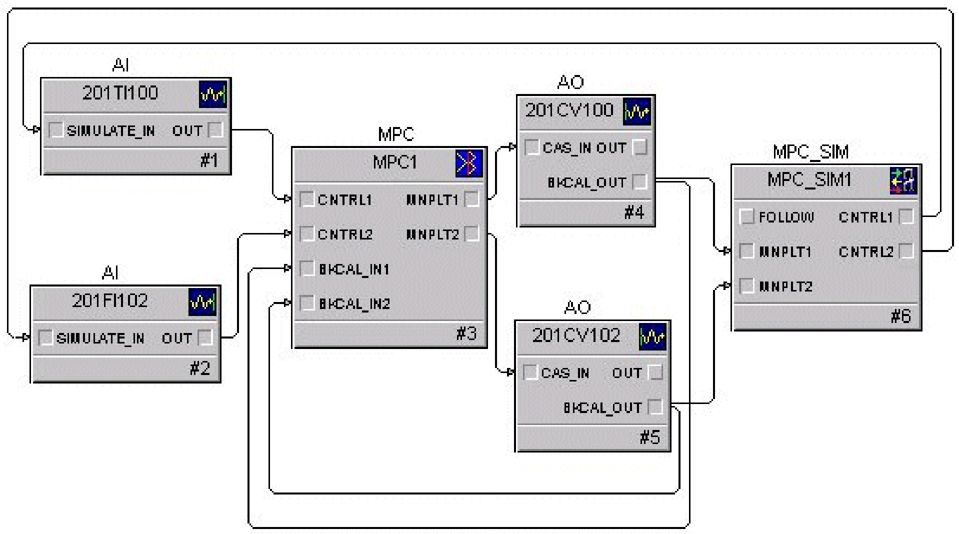
2 Add an MPC Process Simulator function block to this module and change the number of inputs and outputs to match those of the MPC function block.

3 Assign the model used to generate the MPC function block control to the MPC Process Simulator function block. Right-click the MPC function block, select the Predict application, and select File | Assign Simulation Model.

4 Wire the BKCAL_OUT of the blocks downstream of the MPC function block to the manipulated inputs of the MPC Process Simulator function block.

5 Wire the control and constraint outputs of the MPC Process Simulator function block to the SIMULATE_IN of the associated inputs to the MPC function block. Enable Simulate in the analog inputs blocks and set the L_TYPE to Direct Independent.

Once the module containing the MPC Process Simulator function block has been downloaded, the MPC function block should operate similarly to when it is in control of the process. An example of a module containing an MPC Process Simulator function block is shown in the following figure.


AI
201TI100
SIMULATE_IN
OUT
#1
AI
201FI102
SIMULATE_IN
OUT
#2
MPC
MPC1
CNTRL1
MNPLT1
CNTRL2
MNPLT2
BKCAL_IN1
BKCAL_IN2
#3
AO
201CV100
CAS_IN
OUT
BKCAL_OUT
#4
AO
201CV102
CAS_IN
OUT
BKCAL_OUT
#5
MPC_SIM
MPC_SIM1
FOLLOW
CNTRL1
MNPLT1
CNTRL2
MNPLT2
#6

Model Predictive Control Professional (MPCPro) Function Block

The DeltaV Model Predictive Control Professional (MPCPro) function block allows large interactive processes (as large as 20 x20) to be controlled within measurable operating constraints while accounting for process interaction and measurable disturbances. Use the MPCPro block instead of the MPC block if the application size exceeds 8x8 or the application requires more optimization than Constraint pushing (provided by the MPC block). Refer to the Model Predictive Control Function Block topic for more information on the MPCPro block. An embedded optimizer is included with the MPCPro block. The optimizer can be used to effectively provide maximum profit or lowest cost production within the process constraints and limits on process inputs.

The MPCPro function block can replace traditional control systems that use feedforward, decoupling networks, and override for multivariable control. The parameters of the MPCPro function block are configured in Control Studio. When a module that contains the MPCPro block is downloaded, all inputs and outputs used by the MPCPro block are automatically assigned to the Continuous Historian.

Note The history collection rate of the MPCPro function block parameters is defined as part of the block configuration.

You use the DeltaV PredictPro application to commission the MPCPro block, to easily determine the process step response, and to automatically test the process. Using the historic data that reflects automatic or manual process testing, the PredictPro application can automatically identify the process step responses. The control definition used by the MPCPro function block can be generated from the model that is identified. The PredictPro Simulation application can be used to verify the control before it is used in your process. Refer to the PredictPro topic for more information on the PredictPro application.

The operator runs the control through a DeltaV Operate screen built with the MPCPro dynamos. For more information, refer to MPCPro dynamos in the Dynamo Reference topic.

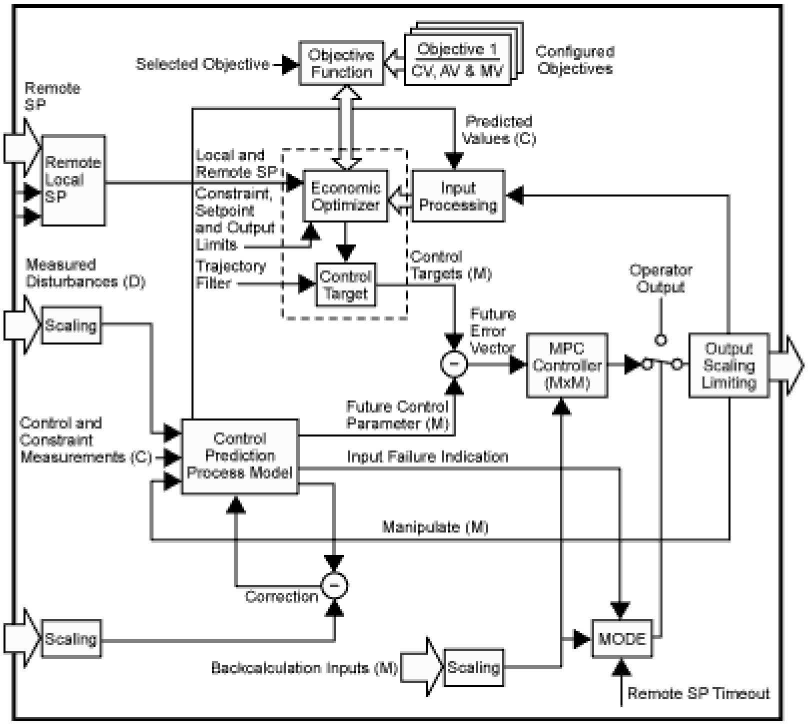
The MPC Simulate Pro application can be used in conjunction with the MPCPro block to create an operator training system offline to demonstrate the control that the MPCPro block can provide.

The MPCPro function block supports modes and status handling.

Note The function blocks that provide the MPCPro inputs and outputs (AI, AO, and Scaler, PID, Ratio, Fuzzy, Bias/Gain) can reside in the same module as the MPCPro block or in other modules. Use the Scaler function block as the MPCPro reference when another function block that does not support scaling (such as a Calc/Logic function block) is referenced as an input to the MPCPro function block. Insert the Scaler function block after the Calc/Logic block and reference the Scaler output in the MPCPro function block. This provides the appropriate parameter (OUT_SCALE) for the input parameters reference by the MPCPro block. Refer to the following figure, MPCPro Function Block, and notice that the MPCPro block has no input and output connections. The user configures the MPCPro block in DeltaV Control Studio to directly reference function block inputs or outputs.

MPCPro Function Block

Schematic Diagram – MPCPro Function Block



MPCPro Function Block Schematic Diagram

Block Execution – MPCPro Function Block

When implementing a multivariable control strategy, the process's Manipulated and Disturbance inputs and Controlled and Constraint outputs are defined in the MPCPro function block. This block can replace conventional implementations involving decoupling, feedforward control, override control, and complex dynamics. In addition, the MPCPro block's economic optimizer can be used to meet a configured objective based on the cost or profit associated with process inputs and outputs.

The target mode of the MPCPro function block and the status of the block’s inputs determine the actual mode of the MPCPro block and the output status. In addition, because of the communication between the downstream function blocks and the MPCPro function block, the previous status of the downstream blocks influences the mode of the MPCPro block.

The MPCPro algorithm executes in the following order:

1. It evaluates the actual mode based on target mode and the status of block inputs.
2. It compares the current value of predicted process Control and Constraint parameters to those calculated by the model and updates the model.
3. It determines the optimum Manipulated parameter values that maximize profit without violating process input or constraint limits (based on the objective selected by the operator). It then calculates the associated value of the process outputs.
4. The MPCPro controller executes using the target value of selected Control and/or Constraint target values.
5. It determines the status and value for block outputs update.

The processing done at each step of execution is described in the following sections.

Evaluate Actual Mode

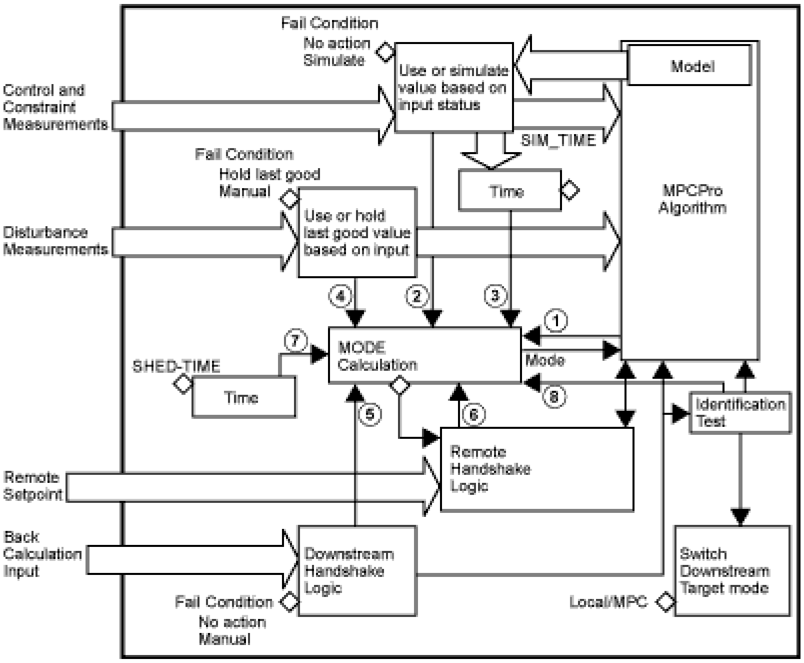
The MPCPro block actual mode is based on the following:

- Target Mode
- Status attribute of the MV back calculation input parameter
- Status attribute of CV, AV, and DV input parameter
- Status of remote setpoint value(s) and time since updated
- Failure Action configured for MV, CV, AV and DV
- STATUS_OPTS selection for treating Uncertain as Good or Bad
- State of MPCPro control generation

Based on these conditions, the actual block mode must be calculated before the MPCPro algorithm executes. The actual mode influences the path taken in the MPCPro algorithm execution.

Mode Evaluation Order

The actual mode of the MPCPro block must be determined after all inputs are read but before the MPCPro control algorithm is executed. The following figure shows the conditions that are determined in the block before the actual mode is calculated.



Block Conditions Before Actual Mode Calculation

The conditions that impact mode are evaluated before executing the MODE calculation. These conditions are evaluated in the following order:

1 **Not Generated –** If the control has not been generated for this block, then the Generate flag is set. Otherwise this flag is cleared.

2 **Bad Control Input**:

- If the status of one or more Control or Constraint parameters is Bad, Good Constant, or Uncertain (and the status option is configured to treat Uncertain as Bad) AND the Fail Condition for that input is Manual, then the Bad Control Input flag is set. Otherwise, the Bad Control Input Flag is cleared.
- If the input is Bad , Uncertain (and treated as Bad) or Good Constant, then take the configured fail action.
- If Simulate is defined as the Fail Condition, or the input has a status of constant (analyzer or manual input) then the associated model value from the last execution is passed to the MPCPro algorithm as the measured input. A Simulate timer is set (if not already set). Reset the Simulate time if none of the Control inputs are being simulated.

3 **Simulate Too Long** – The simulate timer is checked to see if it is set or cleared. If the Simulate timer exceeds the timeout limit based on time to steady state, then the Simulate Too Long flag is set. Otherwise, the Simulate Too Long flag is cleared.

4 **Bad Disturbance Input** – If the status of one or more Disturbance parameters is Bad or Uncertain (and the status option is configured to treat Uncertain as Bad) AND the Fail Condition for that input is Manual, then the Bad Disturbance Input flag is set. Otherwise, the Bad Disturbance Input flag is cleared. If a Disturbance input is Bad and Hold Last Good is defined as the Fail Condition, then the input value used in the MPCPro control is not updated.

5 **Bad Downstream –** If no handshake has completed or the handshake associated with one or more of the Manipulated process inputs is incomplete (as a result of the back calculation inputs having a status of Bad or a status of Good Cascade Not Invited) and the associated Fail Condition for that input is Manual, then set the Bad Downstream flag. Otherwise, the Bad Downstream flag is cleared. If the handshake is incomplete and the Fail Condition is hold last good, then the last good back calculation input is passed as the current position of the MV to the MPCPro algorithm.

6 **Remote Unavailable** – If the target mode of the MPCPro block is RCAS and the handshake with one or more of the RCAS inputs is not complete, then the Remote Unavailable flag is set. Otherwise, the Remote Unavailable flag is cleared.

7 **Remote Timeout** – If the actual mode of the MPCPro block is RCAS, then a remote timer is set (if not already set) for each RCAS input that has not been updated since the last execution. If any of the RCAS timers exceed the limit based on the time to steady state, then the Remote Timeout flag is set. If the actual mode of the block is not RCAS, then the remote timers are reset.

8 **Test Active –** Handshake is complete with downstream loops that will be required for automated testing and process identification is ready to start. While testing is active, the mode of the MPCPro block transitions to LO to indicate to an operator that one or more MPCPro outputs are automatically adjusted.

Modes – MPCPro Function Block

The MPCPro function block supports the following modes:

- **Out of Service** (OOS) – Set by changing the target mode to OOS. In the OOS mode, the block does not execute its control algorithm and the status of all block inputs are set to BAD OUT of Service.
- **Initialization Manual** (IMAN) – Indicates that the path to the process is broken. A common cause is that a required downstream block is not in the correct mode to use the output of the MPCPro block.
- **Local Override** (LO) – Indicates that testing by the PredictPro application is in progress.
- **Manual** (MAN) – Operator has full control of the required MPCPro block outputs.
- **Automatic** (AUTO) – Outputs are automatically calculated by the MPCPro block.
- **Remote Cascade** (RCAS) – The setpoint of Control parameters that are configured for remote setpoint can be changed by another application.

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Mode Calculation

The MPCPro block actual mode is calculated as follows based on the target mode and the condition:

- If the MPCPro Target mode is OOS, then the actual mode is OOS and the status of the MPCPro outputs are set to Bad OOS.
- If the MPCPro Target mode is not OOS and the Test Active flag is set, then the actual mode is LO to indicate that testing is active.
- If the MPCPro Target mode is not OOS and the Bad Downstream flag is set, then the actual mode is IMAN to indicate that the path to the process is broken.
- If the MPCPro Target mode is not OOS and the Bad Downstream flags are clear but the Bad Control or Bad Disturbance flag is set, then the actual mode is MAN.
- If the MPCPro Target mode is AUTO and the Bad Downstream, Not Generated, Bad Control, Bad Disturbance and the Simulate Too Long flags are clear, then the actual mode is AUTO. If the Target mode is MAN for these conditions, then the Actual mode will be MAN.
- If the MPCPro Target mode is AUTO and the Bad Downstream, Not Generated, Bad Control, and Bad Disturbance flags are clear and the Simulate Too Long flag is set, then the Actual mode is MAN. If the Target mode is MAN for these conditions, then the Actual mode will be MAN.
- If the Target mode is RCAS and the Bad Downstream, Not Generated, Bad Control, Bad Disturbance, and Simulate Too Long flags are clear but the Remote Unavailable flag is set, then the Actual mode is AUTO.
- If the Target mode is RCAS and the Bad Downstream, Bad Control and Bad Disturbance, Not Generated, Remote Available, Simulate Too Long and Remote timeout flags are clear then the Actual mode is RCAS.
- If the Target is RCAS and the Bad Downstream, Bad Control and Bad Disturbance, Not Generated, Remote Available, and Simulate Too Long flags are clear and the Remote Timeout flag is set and the previous Actual mode was RCAS, then the block mode should shed to the next available mode.

When the status of a downstream block changes to Good Cascade Initialization Request, then, if the MPCPro block Target mode is not OOS, it will immediately do a handshake with the downstream block. After the handshake completes, the initialization value is maintained in the associated output unless it is changed by an operator (in IMAN or MAN mode) or changed by the MPCPro algorithm (in AUTO or RCAS mode).

Indications of a limit condition or alarm active in the input status provided to MPCPro has no impact on the MPCPro block mode or the handshake that is done by the MPCPro block. An operator disabling a process input or output has no impact.

Update Output Value and Status

The manner in which the MPCPro function block outputs are updated depends on the mode of the MPCPro block and whether a cascade connection has been established with a downstream block.

- In OOS mode, the status of the MPCPro outputs is set to BAD Out of Service. The block does not change the value of block outputs.
- In IMAN mode, the MPCPro function block is unable to establish a cascade connection with one or more of the downstream blocks that are configured as required for control. Normally this occurs when they are not in the correct mode. Until the initialization process is complete for an output, the calculated output provided by the MPCPro algorithm to match the value on its associated BKCAL_IN is written to the block output. Once the initialization process is complete for an input, the output remains at its last value with a status of GOOD Cascade NonSpecific, Not limited.
- In MAN mode, the output values are not written by the MPCPro algorithm but can be changed by the operator. The status of the outputs is GOOD Cascade NonSpecific, Not Limited.
- In AUTO mode, the calculated output is written to the output or maintained at last value if the calculated output will drive the output further within its downstream limit. The output status is set to Good Cascade NonSpecific.

MPCPro Function Block Execution Rate

MPCPro can be used in an application as long as a function block execution time of one second or slower is sufficient to meet your process control requirements. The MPCPro function block execution is automatically set based on the Time to Steady State to provide an algorithm execution rate of one second or slower. Since the MPCPro algorithm uses model and controller matrices that are generated on a very specific time base, the DeltaV PredictPro application automatically configures the effective scan rate of the MPCPro block. The execution time is defined by the process response time estimation (entered as the Time to Steady State when you request that the process be tested for identification) divided by prediction horizon (120 samples) in seconds.

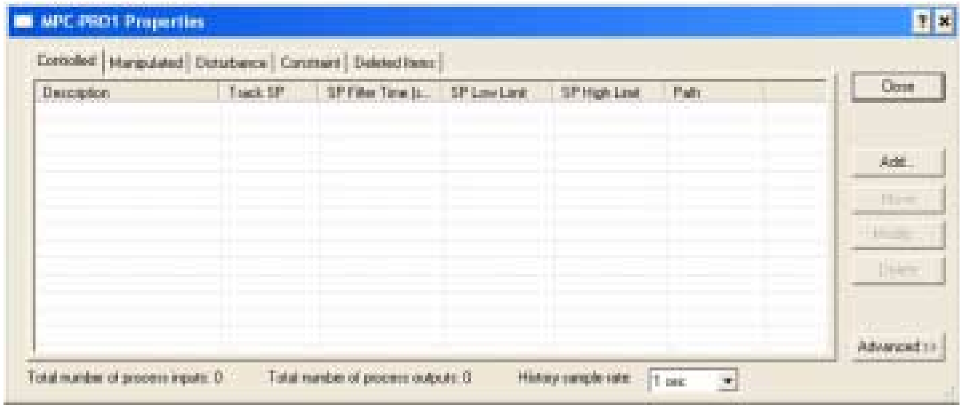
Alarming

When the MPCPro function block detects configuration errors or abnormal conditions, the errors and abnormal conditions are automatically reflected in the BLOCK_ERR parameter. Set the BAD_MASK parameter to determine which condition set BAD_ACTIVE. All conditions not included in BAD_ACTIVE are reflected in the ABNORMAL_ACTIVE parameters. These xxxx_ACTIVE parameters can be used in defining module alarms.

MPCPro Configuration

Note Any change in the block configuration requires that you reconnect to the block from the PredictPro application for the change to be reflected in the block's behavior.

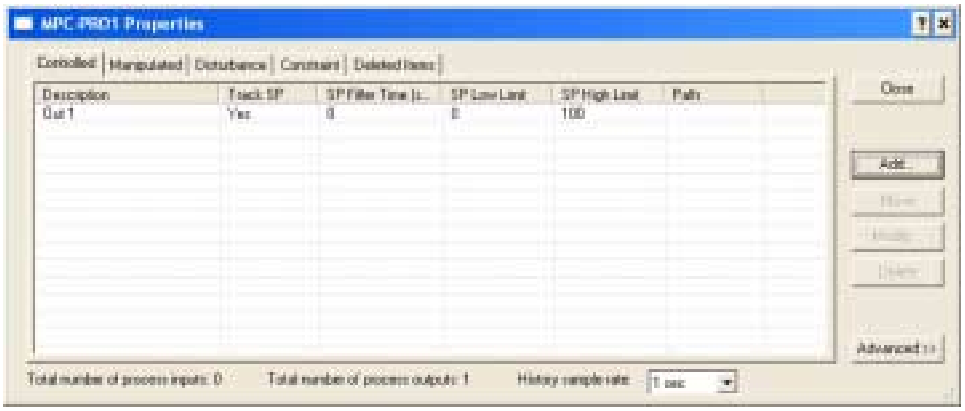
Since the number of inputs and outputs associated with an MPCPro application can vary, the MPCPro block is designed to support a total of twenty (20) Disturbance and Manipulated inputs and a total of twenty (20) Controlled and Constraint parameters. Use the Control Studio application to configure the MPCPro block. To enter configuration information, right-click the MPCPro function block and select Properties to open the Properties dialog box. Use this dialog box to add Control, Manipulate, Constraint, or Disturbance parameters and enter configuration information.



Description	Track SP	SP Filter Time (s...	SP Low Limit	SP High Limit	Path

Total number of process inputs: 0 Total number of process outputs: 0 History sample rate: 1 sec

Select the Controlled, Manipulated, Disturbance or Constraint tab on this dialog and click the Add button to add a new entry. The Controlled tab is selected in the following image.



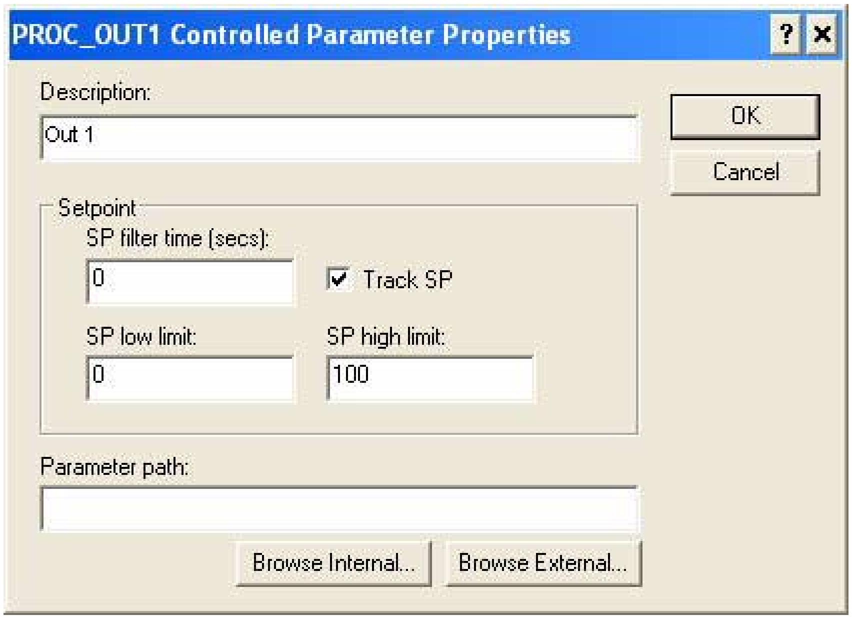
Description	Track SP	SP Filter Time (s...	SP Low Limit	SP High Limit	Path
Out 1	Yes	0	0	100	

Total number of process inputs: 0 Total number of process outputs: 1 History sample rate: 1 sec

Once you have added parameters to the MPCPro block, you can modify them from this dialog by selecting a parameter and choosing Modify. A dialog for defining the associated parameters opens.

Configuring Controlled Parameters

For a Controlled parameter, the following dialog opens.


PROC_OUT1 Controlled Parameter Properties

Description: Out 1

Setpoint

SP filter time (secs): 0

☑ Track SP

SP low limit: 0

SP high limit: 100

Parameter path:

Provide the following configuration information:

Description – The parameter name that you want to appear in the PredictPro application and the MPCPro Operate view. (MPCPro Operate is built with MPCPro dynamos. Refer to MPCPro dynamos in the Dynamo Reference topic for more information.) The default description name is the input instance number for example, Out 1. You can create a new description; it is recommended that the parameter description contain 14 or fewer characters.

SP Filter Time – The filter time (in seconds) is used to set the setpoint trajectory.

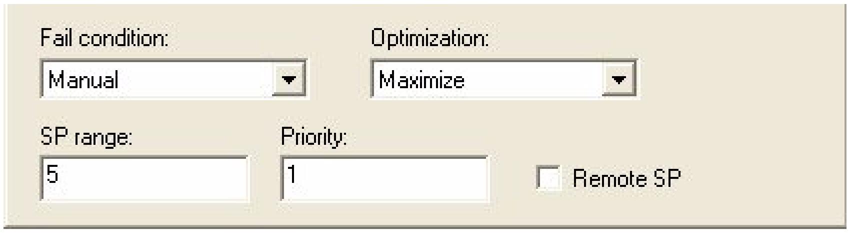
SP Low Limit – The minimum SP value (in engineering units) that an operator can enter. A setpoint entry that is lower than this limit is truncated to the limit value.

SP High Limit – The maximum SP value (in engineering units) that an operator can enter. A setpoint entry that is greater than this limit is truncated to the limit value.

Track SP – If checked, the SP will track the Controlled parameter value when the MPCPro block is in a Target mode of MAN or Actual mode of IMAN .

Parameter path – The reference to the block output that will provide the Control parameter value. If the block providing this value is contained in the same module as the MPCPro block, select the Browse Internal button to define the parameter path. Otherwise, select the Browse External button to define the parameter path.

When the advanced button is selected in the property dialog, additional information appears at the bottom of the Control parameter dialog as shown below.


Fail condition: Manual

Optimization: Maximize

SP range: 5

Priority: 1

☐ Remote SP

Fail Condition - The action to be taken when the input status indicates a BAD input. The actions that can be taken are:

- **No Action** – Take no control action on the measurement value.
- **Shed Local Modes**– Change the downstream blocks associated with the Manipulated parameters to their local mode – as defined by Local Shed mode selection of the Manipulate parameter.
- **Manual** – Change the actual mode of the MPCPro block to Manual until the condition clears.
- **Simulate Fail Manual** – Simulate the measurement input. If the time of simulation exceeds the maximum allowed (based on time to steady state), then the MPC actual mode will change to Man until this condition clears.
- **Simulate Fail Shed Local** – Simulate the measurement input. If the time of simulation exceed the maximum allowed (based on time to steady state), then the downstream blocks associated with the Manipulated parameters will be forced to their local mode – as defined by Local Shed mode selection of the Manipulate parameter.

Optimization –The impact of the Control parameter on plant cost or profit. The selections are:

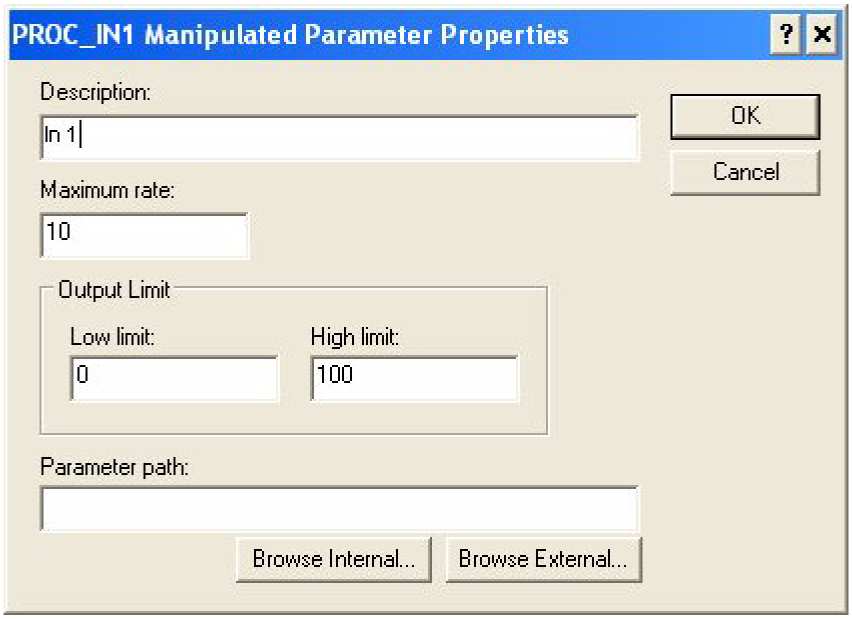
- **None** – Not considered in the objective function. There is no cost or profit associated with this parameter.
- **Maximize** – Indicates that this parameter is associated with profit.
- **Minimize** – Indicates that this parameter is associated with cost.

SP range–The maximum amount that the optimizer can change the working setpoint used in control from the operator specified setpoint.

Priority –Numeric value that can be assigned to Control and Constraint parameters to indicate relative importance where the smaller number indicates greater importance/priority. If it is not possible for the optimizer to find a solution that satisfies all operating constraint limits within the specified control ranges, then a solution will be found that satisfies the control ranges and constraint limits that have the highest priority.

Remote SP –The setpoint of the Control parameter will be set by another application or function block that is writing to the associated RCAS_IN(xx) parameter.

Configuring Manipulated Parameters
For a Manipulated parameter, the following dialog opens.


PROC_IN1 Manipulated Parameter Properties

Description: In 1

Maximum rate: 10

Output Limit

Low limit: 0

High limit: 100

Parameter path:

Provide the following configuration information:

Description – The parameter name that you want to appear in the PredictPro application and the MPCPro Operate view. (MPCPro Operate is built with MPCPro dynamos. Refer to MPCPro dynamos in the Dynamo Reference topic for more information.) The default description name is the input instance number for example, In 1. You can create a new description; however, it is recommended that the parameter description contain 14 or fewer characters.

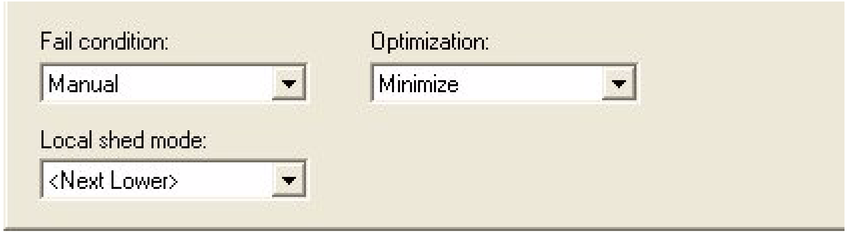
Maximum MV Rate – The maximum rate (in engineering units per second) at which the MPCPro function block can change the Manipulated parameter.

Low Limit – The minimum output value (in engineering units) that an operator can enter. An entry that is lower than this limit is truncated to the limit value.

High Limit – The maximum output value (in engineering units) that an operator can enter. An entry that is greater than this limit is truncated to the limit value.

Parameter Path – The reference to the block input that is written to the manipulate value. Based on this path, the MPCPro block automatically accesses the mode and back calculation output of the downstream block. If the block providing this value is contained in the same module as the MPCPro block, select the Browse Internal button to define the parameter path. Otherwise, select the Browse External button to define the parameter path.

When the advanced button is selected in the Property dialog, the additional information appears at the bottom of the Manipulate parameter dialog as shown in the following figure.



Fail condition: Manual

Optimization: Minimize

Local shed mode: <Next Lower>

Fail Condition – The action to be taken when the feedback from the downstream block contains a status of BAD or Not Invited:

- **No Action** – Maintain the manipulate parameter at its current value with no impact on MPCPro mode.
- **Shed Local Modes** – Change the downstream blocks associated with the Manipulated parameters to their local mode – as defined by Local Shed mode selection of the Manipulate parameter.
- **Manual** – Change the actual mode of the MPCPro block to MAN until the condition clears.

Optimization – The impact of the manipulate parameter on plant cost or profit. The selections are:

- **None** – Not considered in the objective function. There is no cost or profit associated with this parameter
- **Maximize** – Indicates that this parameter is associated with profit
- **Minimize** – Indicates that this parameter is associated with cost.

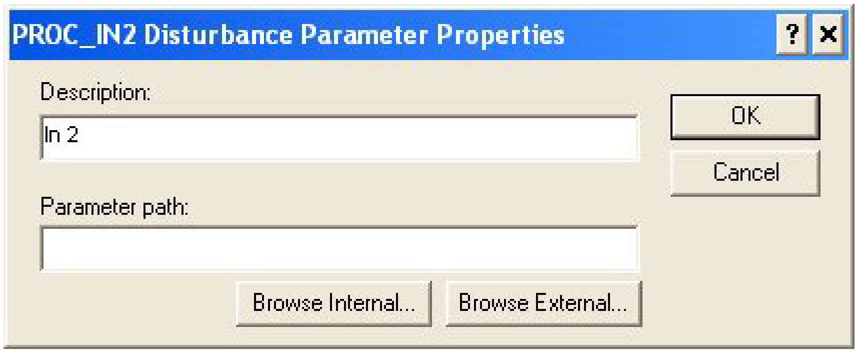
Local Shed Mode –The target mode that the downstream block will be forced to when a input or output failure is detected and the Failure action is defined as Shed Local. The selections are:

- **Next Lower –**The mode will be written to the next lower permitted mode of the downstream block. The selection order is:

=>Rout =>Rcas =>Cas >=>Auto =>Man

Configuring Disturbance Parameters

For a Disturbance parameter, the following configuration dialog appears.



Description: In 2

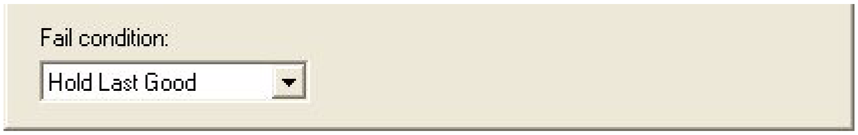
Parameter path:

Provide the following configuration information:

Description – The parameter name that you want to appear in the PredictPro application and the MPCPro Operate view (MPCPro Operate is built with MPCPro dynamos. Refer to MPCPro dynamos in the Dynamo Reference topic for more information.) The default description name is the input instance number for example, In 2. You can create a new description; however, it is recommended that the parameter description contain 14 or fewer characters.

Parameter path – The reference to the block output that provides the Disturbance input value. If the block providing this value is contained in the same module as the MPCPro block, select the Browse Internal button to define the parameter path. Otherwise, select the Browse External button to define the parameter path.

When the advanced button is selected in the Property dialog, the additional information appears at the bottom of the Disturbance parameter dialog as shown in the following figure.



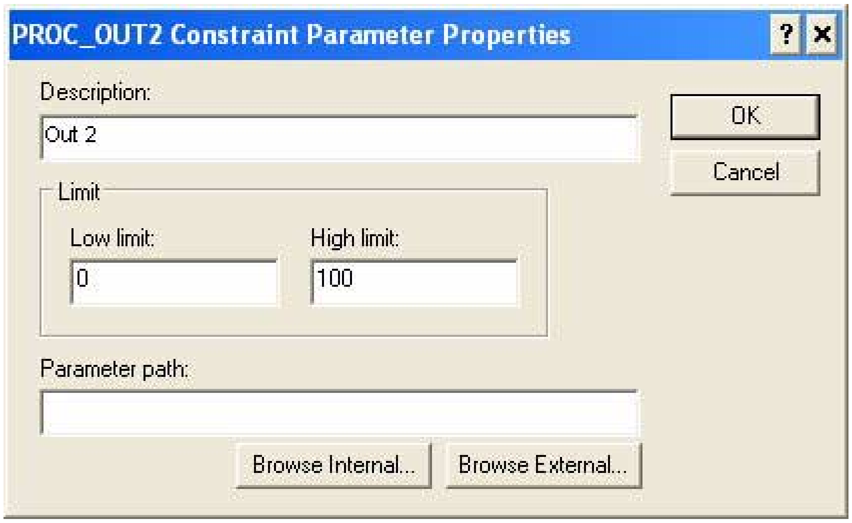
Fail condition: Hold Last Good

Fail condition: the action that should be taken when the Disturbance input has a BAD status. One of the following actions can be selected:

- **Hold Last Good** – Maintain the last value of the Disturbance input that had a status of Good.
- **Shed Local Modes** – Change the downstream blocks associated with the Manipulated parameters to their local mode as defined by the Local Shed Mode selection of the manipulate parameter.
- **Manual** – Change the actual mode of the MPCPro block to manual until the condition clears.

Configuring Constraint Parameters

For a Constraint parameter, the following configuration dialog appears.



Description: Out 2

Low limit: 0

High limit: 100

Parameter path:

Provide the following configuration information:

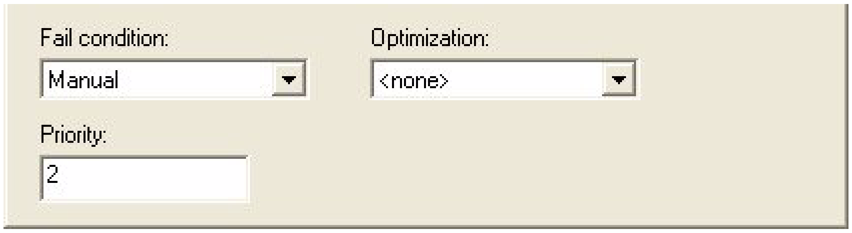
Description – The parameter name that you want to appear in the PredictPro application and the MPCPro Operate view (MPCPro Operate is built with MPCPro dynamos. Refer to MPCPro dynamos in the Dynamo Reference topic for more information.) The default description name is the input instance number, for example, 2nd output. You can create a new description; however, it is recommended that the parameter description contain 14 or fewer characters.

Low Limit – The minimum Constraint target (in engineering units).

High Limit – The maximum Constraint target (in engineering units).

Parameter path – The reference to the block output that provides the Constraint input value. If the block providing this value is contained in the same module as the MPCPro block, select the Browse Internal button to define the parameter path. Otherwise, select the Browse External button to define the parameter path.

When the advanced button is selected in the property dialog, the additional information appears at the bottom of the Constraint parameter dialog as shown in the following figure.



Fail condition: Manual

Optimization: <none>

Priority: 2

Fail Condition – The action to be taken when the input status indicates a BAD input. The actions that can be taken are:

- **No Action**– Take no control action on the measurement value.
- **Shed Local Modes**– Change the downstream blocks associated with the Manipulated parameters to their local mode – as defined by Local Shed mode selection of the Manipulate parameter.
- **Manual**– Change the actual mode of the MPCPro block to Manual until the condition clears.
- **Simulate Fail Manual** – Simulate the measurement input. If the time of simulation exceeds the maximum allowed (based on time to steady state), then the MPC actual mode will change to Man until this condition clears.
- **Simulate Fail Shed Local** – Simulate the measurement input. If the time of simulation exceed the maximum allowed (based on time to steady state), then the downstream blocks associated with the Manipulated parameters will be forced to their local mode – as defined by Local Shed mode selection of the Manipulate parameter.

Optimization – The impact of the Control parameter on plant cost or profit. The selections are:

- **None** – The parameter is not included in the objective function. There is no cost or profit associated with the parameter.
- **Maximize** – Indicates that this parameter is associated with profit.
- **Minimize** – Indicates that this parameter is associated with cost.

Priority – Numeric value that can be assigned to Control and Constraint parameters to indicate relative importance where the smaller number indicates greater importance/priority. If it is not possible for the optimizer to find a solution that satisfies all operating constraint limits within the specified control ranges, then a solution will be found that satisfies the control ranges and constraint limits that have the highest priority.

Parameters Referenced by the MPCPro Function Block

The MPCPro function block can reference input and output blocks assigned to traditional or HART transmitters and valves. It can also reference blocks assigned to fieldbus devices. All function blocks referenced by the MPCPro function block must contain a scaling parameter. During MPCPro execution, this scaling information is automatically read by the MPCPro block and used in input-output scaling within the block. If an MPCPro block's input references a block that does not contain a scaling parameter such as a math block, a Scaler function block is required. Wire the Scaler function block directly to the upstream function blocks and reference the OUT parameter such that the MPCPro block can access the scaling information.

Note The engineering unit span configured for measurements directly impact the step response gain. If this gain is less than 0.1 or greater than 20, then it is assumed that the gain is zero and a flat line response will be shown. Thus, if the measurement operating range is small compared to the engineering unit span of the AI block, then a scaling block should be used after the AI block where both the IN_SCALE and OUT_SCALE are set equal to the normal operating range of the measurement. That is, there is no change in value but the scale as read by the MPCPro block will reflect the normal operating range.

For example, if a temperature measurement has an engineering unit span of 1000 degF but the normal operating range is 520-540, then a Scaler function block could be used with the IN_SCALE and OUT_SCALE set to 500-600. As a result, the calculated step response gain would be greater than 0.1.

Similarly, if small changes in a manipulate parameter have a large impact on Control or Constraint parameters, then a scaler can be used on the input with a range that is greater than the AI blocks so that the gain will be less than 20.

The function blocks that are referenced by the MPCPro block can be within the same module or in a different module. The Manipulated outputs must reference the CAS_IN, RCAS_IN, or ROUT_IN inputs of the downstream function blocks. If the MPCPro block is being used to replace a PID block, it typically references the CAS_IN or RCAS_IN of an AO block. However, for many other applications, the output of the MPCPro function block can reference the cascade input or remote cascade input of a PID, Fuzzy Logic or Ratio function block that provides the process input. In such cases, the MPCPro function block treats the downstream process setpoint as the Manipulated process input. As such, the MPCPro function block's Manipulated output is referenced to the CAS_IN of this function block. The BKCAL_OUT and MODE of the downstream block referenced by the MPCPro Manipulated output is automatically read by the MPCPro block based on the path definition for the manipulated parameter.

In some cases, the MPCPro function block replaces an existing control strategy that uses the CAS_IN of the downstream block. You can layer the MPCPro function block on the existing strategy to allow the MPCPro control to be implemented with the old strategy as a backup. You can achieve this by referencing the RCAS_IN and RCAS_OUT (or the ROUT_IN and ROUT_OUT unless referencing an AO block) of the downstream block. The MPCPro function block can reference these parameters in much the same way that it references the CAS_IN of a block.

Status Handling – MPCPro Function Block

If the status of a Controlled, Constraint, or Disturbance input to the MPCPro block that is Bad, the actual mode can automatically go to Man or continue in Auto using the simulated value or the bad value. The behavior is determined by the Failure condition configured for that input. UNCERTAIN status on inputs are treated as Good if the MPCPro parameter STATUS_OPTS is configured to Use Uncertain as Good.

The BKCAL_OUT, RCAS_OUT or ROUT_OUT status of the downstream blocks are used to handshake with the downstream block for bumpless transfer and to determine if the downstream blocks are in the correct mode in response to changes made by the MPCPro function block. The MPCPro function block’s actual mode remains in IMAN until all required downstream blocks (as determined by the manipulate Fail condition) are in the correct mode and the handshake for bumpless transfer is complete.

If the MPCPro function block's mode is changed to Out of Service, the status of the block outputs change to BAD Out of Service.

The MPCPro function block remains in IMAN until all required downstream blocks (as determined by the manipulate Fail condition) are in the correct mode (CAS, RCAS, or ROUT). The Actual mode of the MPCPro block goes to LO when the PredictPro application is automatically adjusting the manipulated block parameters to test the process.

Parameters – MPCPro Function Block

The following table lists the MPCPro function block parameters that are displayed in Control Studio after the block is configured. Save and re-open the module and then select the block to view these parameters.

MPCPro Function Block Parameters

Parameter	Unit	Description
ABNORM_ACTIVE	None	The indication that a block error condition associated with the MPCPro block not selected in BAD_MASK is True (Active).
BAD_ACTIVE	None	The indication that a block error condition associated with the MPCPro block selected in BAD_MASK is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions is True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions not included in BAD_MASK is True, ABNORM_ACTIVE becomes True.
BKCAL_IN [1…N]	EU of associated manipulated parameter.	The backcalculation value and status (BKCAL_OUT, RCAS_OUT, or ROUT_OUT) from a process input block that is Manipulated by the MPCPro block. This value and status are used by the MPCPro block for bumpless transfer and to determine active limit conditions and whether the downstream block is in the correct mode. The number shown as part of the BKCAL_IN parameter name corresponds to the number contained in the Name field for the associated Manipulated parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Disturbance parameter to a Manipulate parameter, a Name (and BKCAL_IN) is assigned automatically to all N process inputs (Disturbance and Manipulate) even though the BKCAL_IN parameter is used only for Manipulated process input parameters.
BLOCK_ERR	None	The actual error status of the function block.
INSPECT ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 if the following conditions are True: - The Write To Inspect Alarm context menu item was selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. **Note** An abnormal condition will exist for Variability only if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.
MODE	None	Shows and sets the block operating state. MODE contains the actual, target, permitted, and normal modes.

Parameter	Unit	Description
PROC_IN [1…N]	The output units of the connected block.	Shows the value and status of Disturbance and Manipulated parameters, such as the inputs to the process, read by the MPCPro block. The number shown as part of the PROC_IN parameter name corresponds to the number contained in the Name field for the associated Disturbance and Manipulate parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Disturbance parameter to a Manipulate parameter, a Name is assigned automatically to all N process inputs (Disturbance and Manipulate). This number, associated with a process input, remains the same even when the Move option is used to change its use in the MPCPro block.
PROC_OUT [1…M]	The output units of the connected block.	Shows the value and status of Control and Constraint parameters, such as the outputs of the process read by the MPCPro block. The number shown as part of the PROC_OUT parameter name corresponds to the number contained in the Name field for the associated Control or Constraint parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Control parameter to a Constraint parameter, a Name is assigned automatically to all M process outputs (Control and Constraint parameters). This number, associated with a process output, remains the same even when the Move option is used to change its use in the MPCPro block.
RCAS_IN [1…M]	The units of the associated control parameters.	Shows the value and status of the remote setpoints provided by another function block or application. These will be used as the MPCPro block setpoint when the actual mode of the MPCPro block is RCAS. The number shown as part of the RCAS_IN parameter name corresponds to the number contained in the Name field for the associated Control parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Control parameter to a Constraint parameter, a Name is assigned automatically to all M process outputs (Control and Constraint parameters). This number, associated with a process output, remains the same even when the Move option is used to change its use in the MPCPro block.

Parameter	Unit	Description
RCAS_OUT [1…M]	The units of the associated control parameters.	Shows the backcalculation value and status provided to another function block or application that is writing a remote setpoint to the RCAS_IN parameter. The number shown as part of the RCAS_OUT parameter name corresponds to the number contained in the Name field for the associated Control parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Control parameter to a Constraint parameter, a Name is assigned automatically to all M process outputs (Control and Constraint parameters). This number, associated with a process output, remains the same even when the Move option is used to change its use in the MPCPro block.
SP [1…M]	The units of the associated process output parameters.	Shows the setpoints used by the MPCPro block. This value can be set by an operator (in Auto mode) or indirectly by a function block or application when the mode is RCAS (and remote setpoint capability has been selected for the Control parameter). The number shown as part of the SP parameter name corresponds to the number contained in the Name field for the associated Control parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Control parameter to a Constraint parameter (using move selection), a Name is assigned automatically to all M process outputs (Control and Constraint parameters). This number, associated with a process output, remains the same even when the Move option is used to change its use in the MPCPro block.
SP_WRK [1…M]	The units of the associated process output parameters.	Shows the working setpoints used by the MPCPro block. This value can differ from the operator setpoint on a setpoint change based on the setpoint filter. Also, the optimizer can change the working setpoint within the control range to meet a selected objective. The number shown as part of the SP_WRK parameter name corresponds to the number contained in the Name field for the associated Control parameter in the MPCPro Properties dialog when the advanced tab is selected. **Note** Since the Move option is provided to change a Control parameter to a Constraint parameter, a Name is assigned automatically to all M process outputs (Control and Constraint parameters). This number, associated with a process output, remains the same even when the Move option is used to change its use in the MPCPro block.
STATUS_OPTS	None	Option to allow action on Uncertain input to be selected.
STDEV	Percent of scale	The largest standard deviation calculation for the control parameters over the standard deviation time. Refer to Loop Performance Calculations for more details on how this parameter is calculated.

Parameter	Unit	Description
STDEV_CAP	Percent of scale	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.

Application Examples - MPCPro Function Block

The MPCPro function block provided by DeltaV PredictPro can be used to address multivariable control requirements. You can use the PredictPro application to create process models and control definitions from process data. The automated test feature of the PredictPro application allows the step response of Controlled or Constraint outputs to be determined based on changes in the Manipulated inputs. This capability can meet multivariable control requirements in all process industries. In many cases, the MPCPro function block can replace control techniques that have traditionally been used to address the control requirements of multivariable processes.

The implementation and commissioning of the MPCPro function block are far simpler and faster than traditional techniques. For an example of traditional techniques that can be replaced by the MPC and MPCPro block, refer to the Model Predictive Control (MPC) Function Block topic.

Larger and more complex applications that can benefit from MPCPro control can be identified based on your process knowledge. Use the following guidelines to determine when you should consider the MPCPro block:

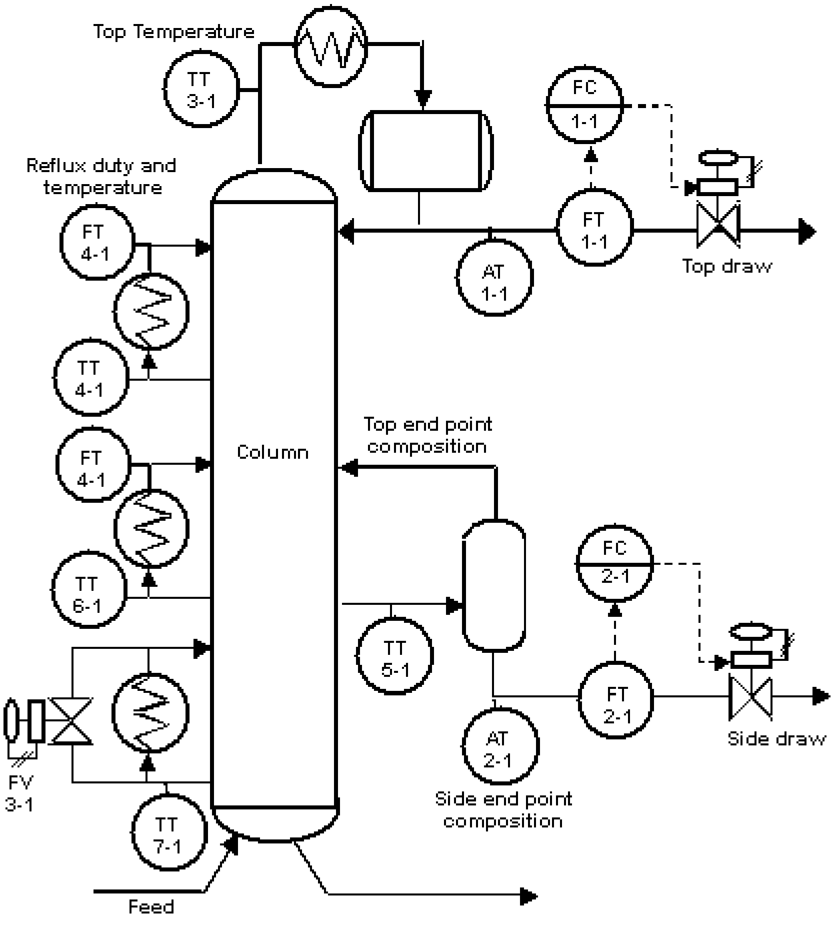
- The number of interactive Control parameters, Constraints or Disturbances exceed the capability of the MPC block.
- The number of Manipulated parameters exceeds the number of Controlled parameters. For example, many of the process outputs can be defined as Constraint parameters.
- You wish to maximize operating profit automatically based on the cost of feedstock and profit of final outputs. Based on this cost information and the process input and Constraint limits, the optimization capability of the MPCPro block can be used to maximum operating profit.
- Feedstock cost are significant and there is some flexibility in the way the product specification can be achieved.

The execution rate of the MPCPro function block is limited to one second or slower. Therefore, MPCPro should be applied to processes where control requirements can be satisfied at these execution speeds.

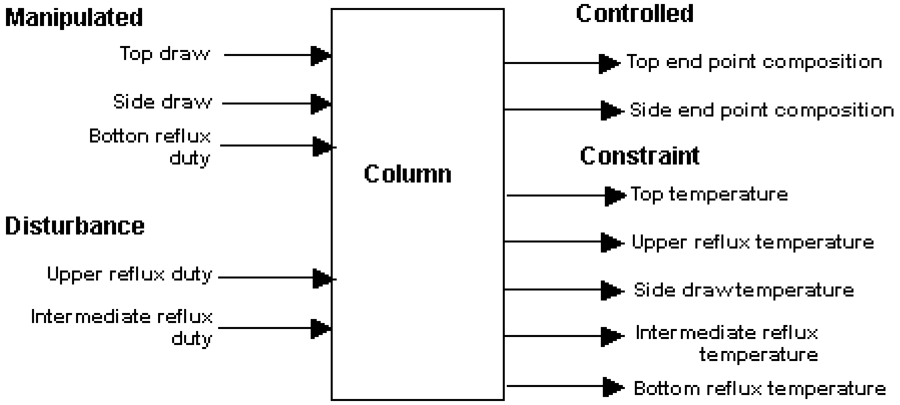
Large Multivariable Applications

The design of many processes requires that the control strategy address multiple Controlled, Constraint, and Disturbance outputs through the adjustment of multiple Manipulated process inputs. The MPCPro function block can control processes of size 20x20, for example, a total of 20 Disturbance and Manipulated parameters, and a total of 20 Control and Constraint parameters.

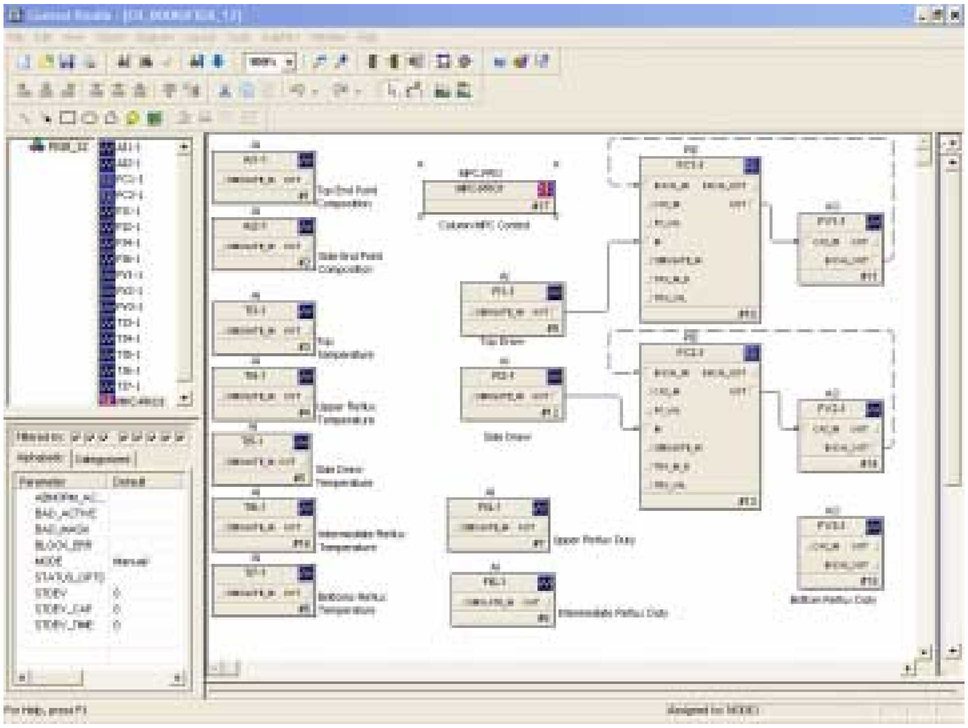
To show how MPCPro can be applied to the control of a multivariable process, we will consider as an example, an oil fractionator process. This problem has been published by Shell and includes typical process dynamics with normalized inputs and output. In this example, it is assumed that the heat requirement of the column enters with the feed. The column has three product draws and three side circulating loops that remove heat to achieve the desired product separation. The heat duty associated with the bottom loop can be adjusted. The heat duty of the other recirculating loop can be measured but is determined by other parts of the plant. The specification for the top and side draws are determined by economics and operating requirements. There is no product specification for the bottom draw but there is an operating constraint on the temperature in the lower part of the column. The following figure shows the process.



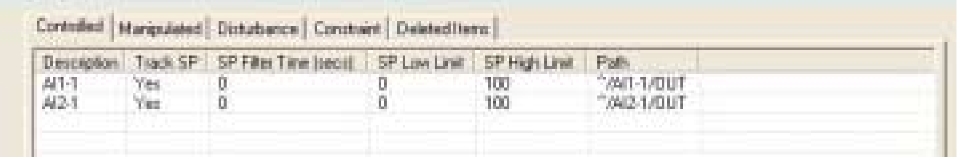
From a control perspective, the process inputs and outputs can be visualized in the following manner based on the process design, available measurements, and operating requirements.



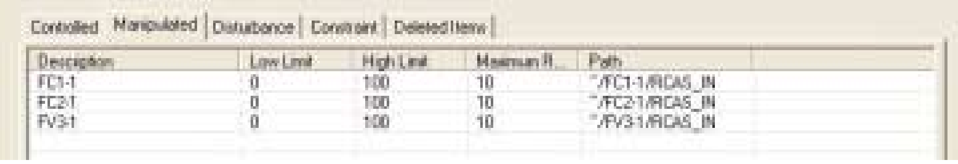
In general, it is recommended that the MPCPro block be included in a module by itself. This allows the module to be downloaded at any time. The MPCPro block can externally reference the measurements and control loops in other control loops. For simplicity, the MPCPro block has been included in the same modules as the measurements and loops it references, as shown in the following figure.



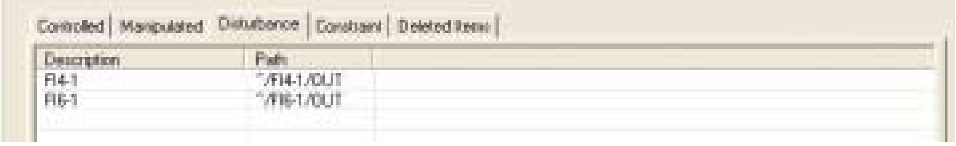
The complete configuration of the MPCPro block is shown in the following figures. The setpoint limits are based on the normalized range in this example. In an actual application, these limits should be set in engineering units based on the actual range of the measurements and actuators.



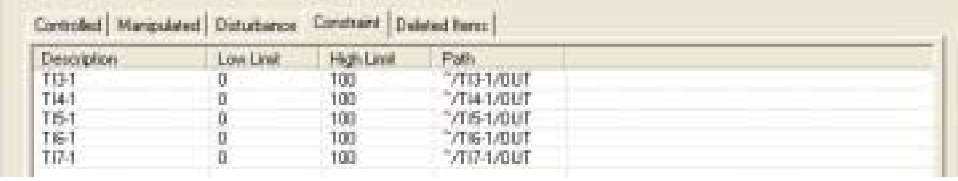
Description	Track SP	SP Filter Time (secs)	SP Low Limit	SP High Limit	Path
AI1-1	Yes	0	0	100	^/AI1-1/OUT
AI2-1	Yes	0	0	100	^/AI2-1/OUT



Description	Low Limit	High Limit	Maximum R...	Path
FC1-1	0	100	10	^/FC1-1/RCAS_IN
FC2-1	0	100	10	^/FC2-1/RCAS_IN
FV3-1	0	100	10	^/FV3-1/RCAS_IN

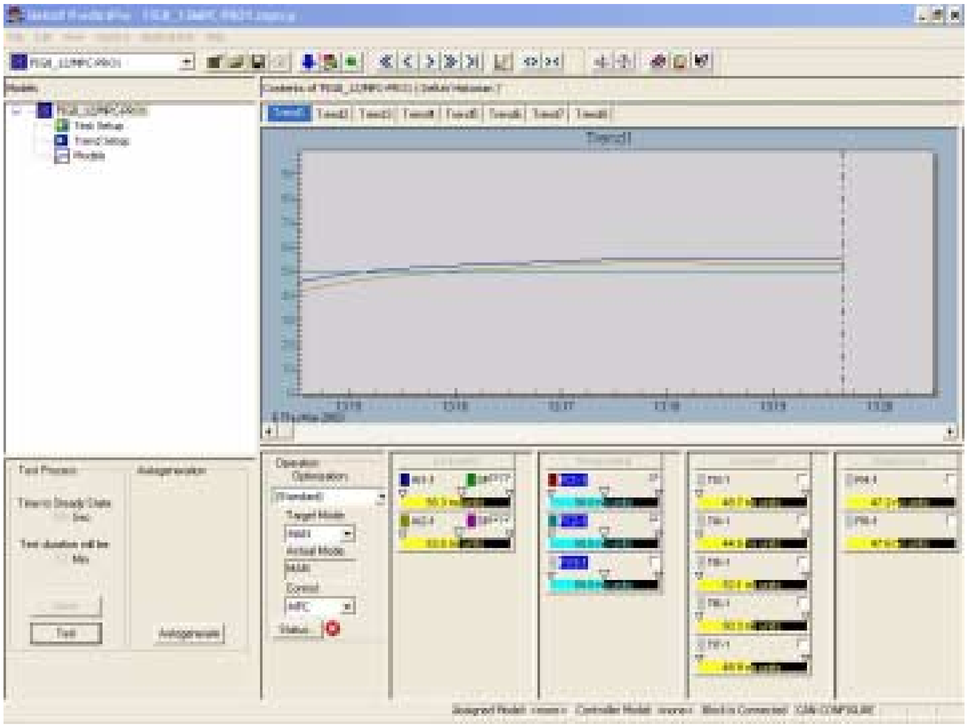


Description	Path
FI4-1	^/FI4-1/OUT
FI6-1	^/FI6-1/OUT



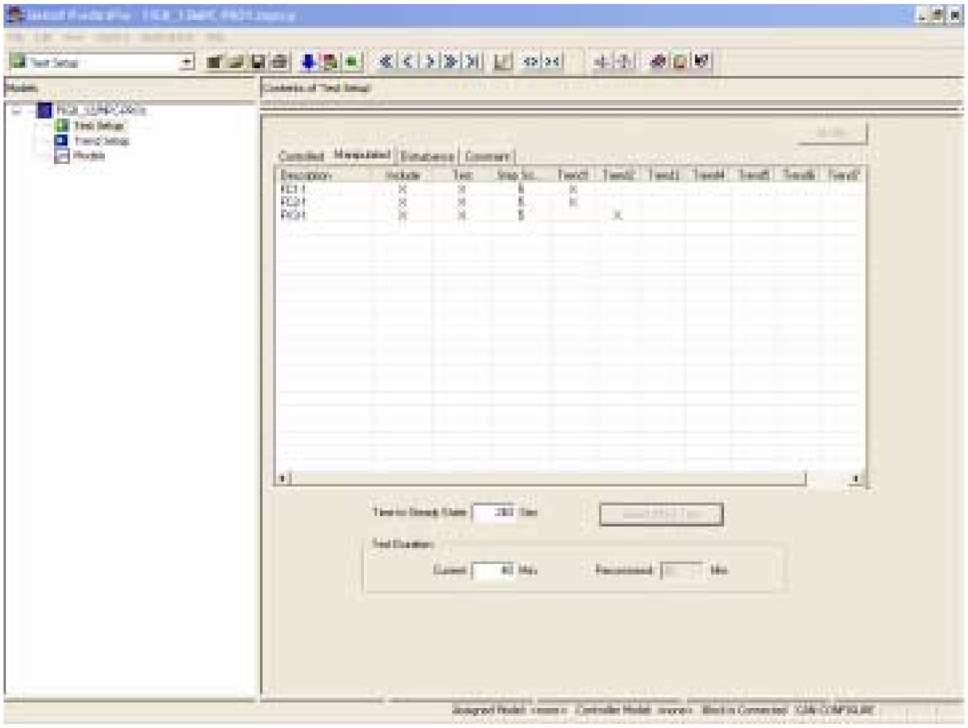
Description	Low Limit	High Limit	Path
TI3-1	0	100	^/TI3-1/OUT
TI4-1	0	100	^/TI4-1/OUT
TI5-1	0	100	^/TI5-1/OUT
TI6-1	0	100	^/TI6-1/OUT
TI7-1	0	100	^/TI7-1/OUT

After this module is downloaded, launch the DeltaV PredictPro application to commission and test the control. The PredictPro application can be launched from the Start menu or by right clicking the MPCPro block The following figure shows the main screen of the PredictPro application as launched from the example MPCPro block.



Based on the MPCPro block configuration, the Control, Manipulated, Constraint, and Disturbance parameters are automatically displayed. Notice that the Description that was configured for each of these parameters is used in the faceplate representation. For this example, the Control selection has already been changed from Local to MPC which forced the downstream blocks to automatically change their target mode to RCAS. Thus, the mode of the MPCPro block is MAN. Also, notice that some parameters are being trended. This is accomplished by clicking in the faceplate check box for the parameter to be plotted.

By selecting the setup button, the Manipulated parameters that are included in the test can be selected, as shown below.



Description	Include	Test	Step Si...	Trend1	Trend2	Trend3	Trend4	Trend5	Trend6	Trend7
FC11	X	X	5	X						
FC21	X	X	5	X						
FIC21	X	X	5		X					

Time to Steady State 280 Sec

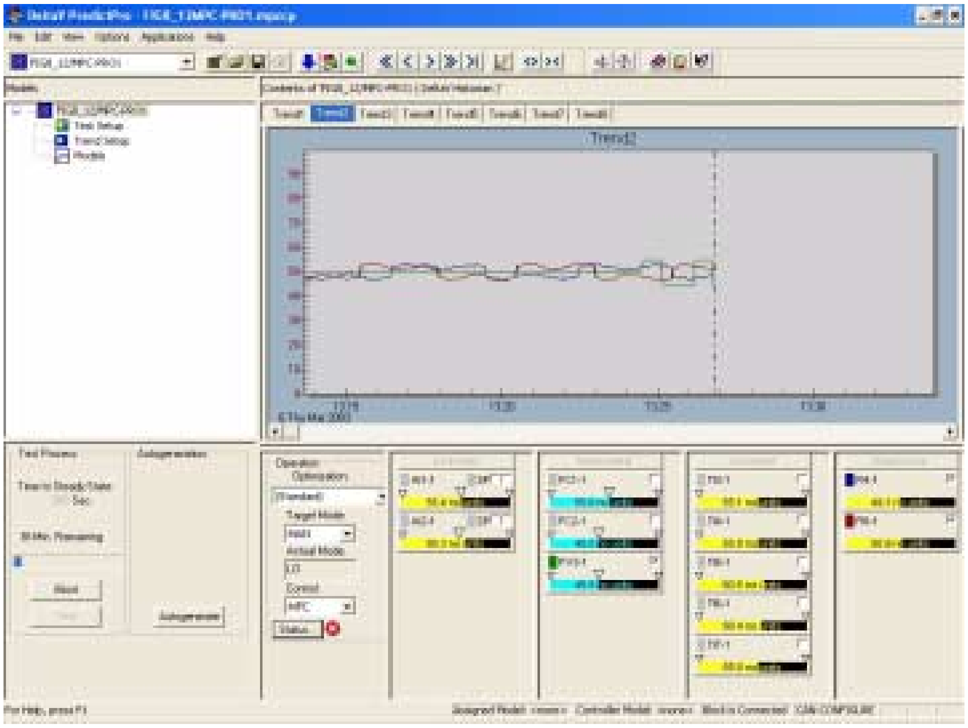
Test Duration

Current 40 Min Recommended [illegible] Min

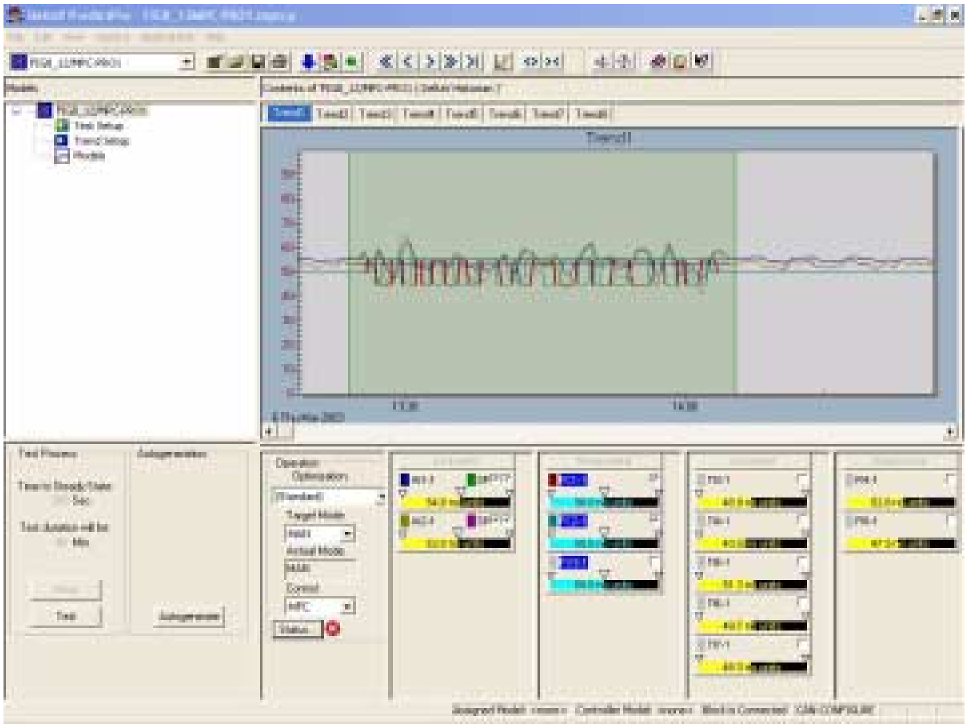
Assigned Model: <none> Controller Model: <none> Block is Connected: CAN CONFIGURE

In this example, all the Manipulated parameters have been chosen for use in the automated test. Click the Select All for Test button to include all parameters. The default step size of 5 percent will be used in the test. Since all process outputs and disturbances are to be identified (the default selection), then nothing more must be done in the setup.

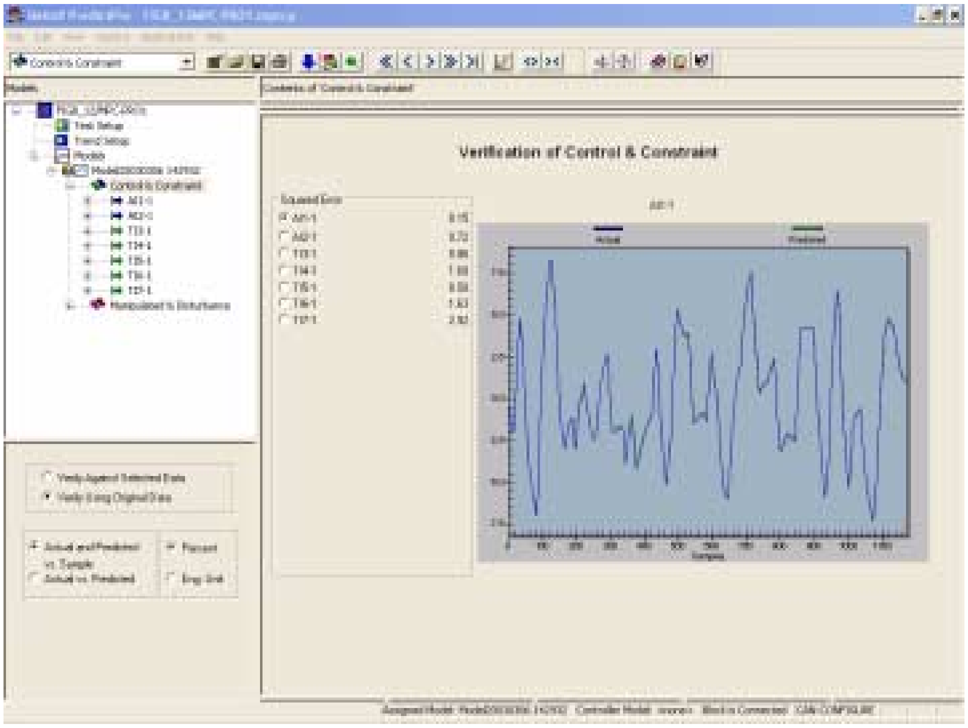
Return to the application's top level view, and click the Test button to run the test . In response, the Manipulated parameters will be automatically changed in a pseudo-random manner.



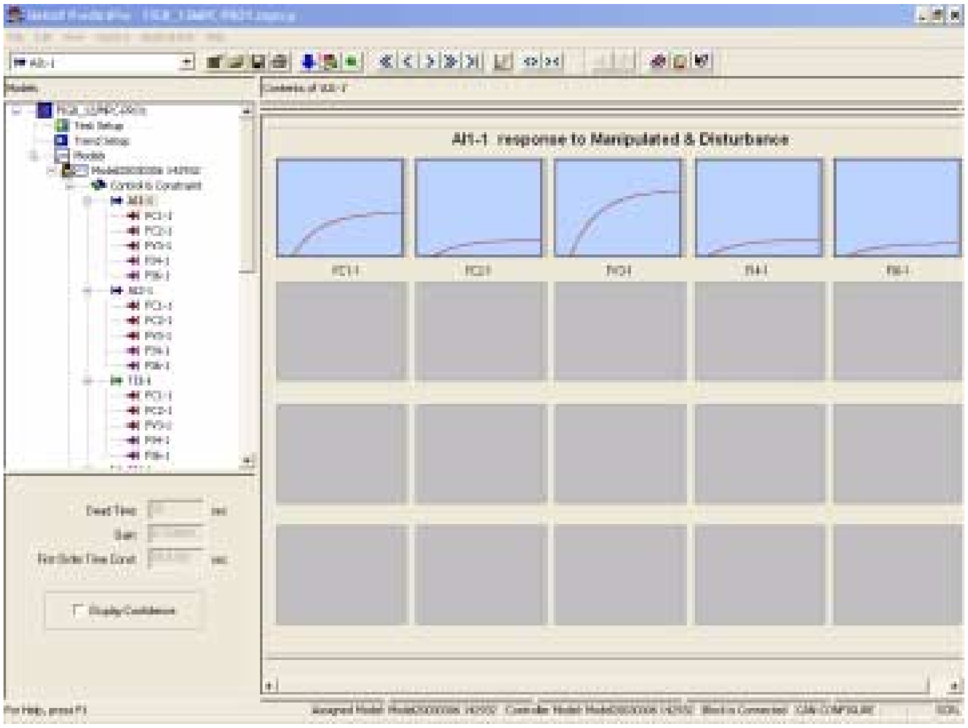
When the automated test is complete, the area of testing is automatically selected as shown in the following figure.



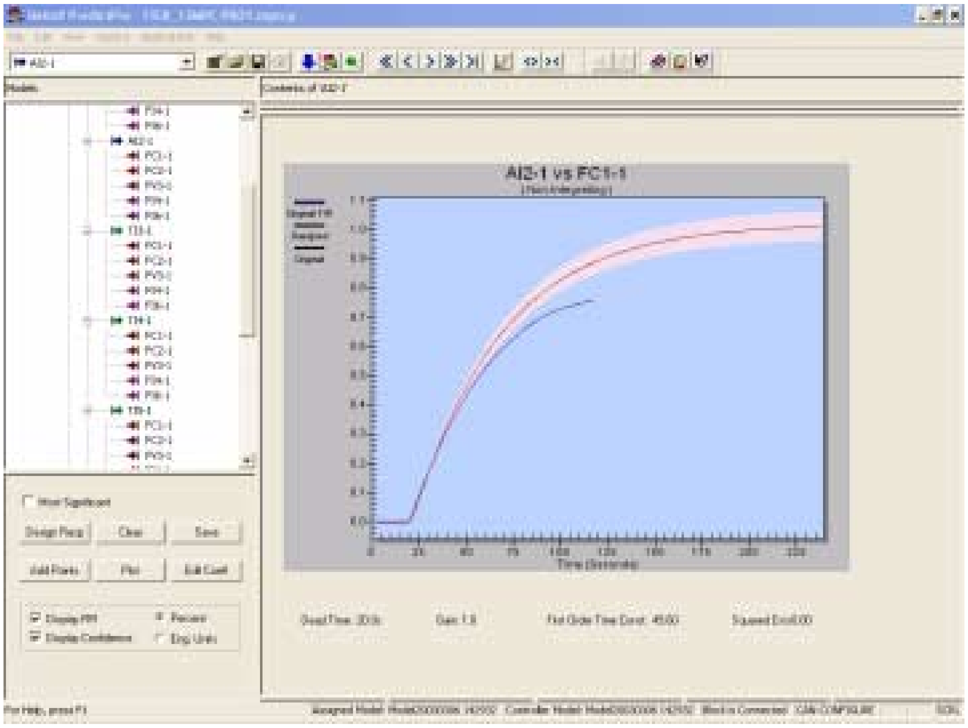
Since the process operation during the time of test was normal, all the test data will be used to generate the model. Thus, by selecting the Autogeneration button, the model and controller for the process is generated. Verification of the set response is shown below. For purposes of this example, the process response has been scaled by a factor of 100.



Based on the square error indicated in the verification, the model has been accurately identified. Further verification of the model can be done by examining each step response and comparing this to your knowledge of the process. An example of the Top End Point Composition is shown in the following figure.



Also, further confirmation of the model can be done by comparing the FIR response to the ARX response and examining the confidence interval for one of the step responses as shown in the following figure. Choose Options | Expert to use this command.



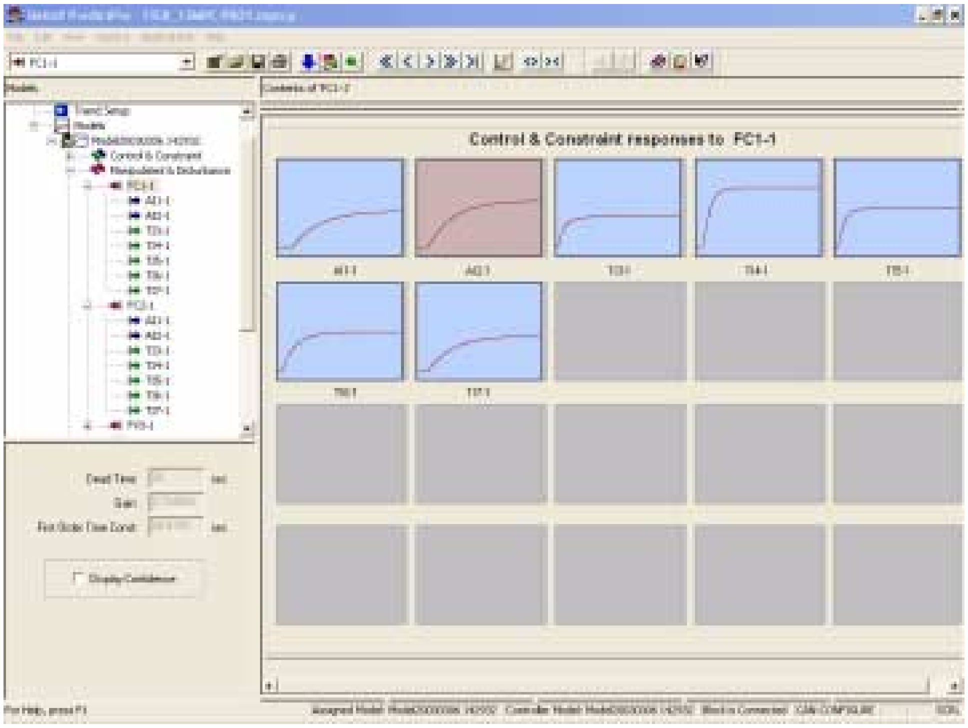
Dead Time: 20.0s

Gain: 1.0

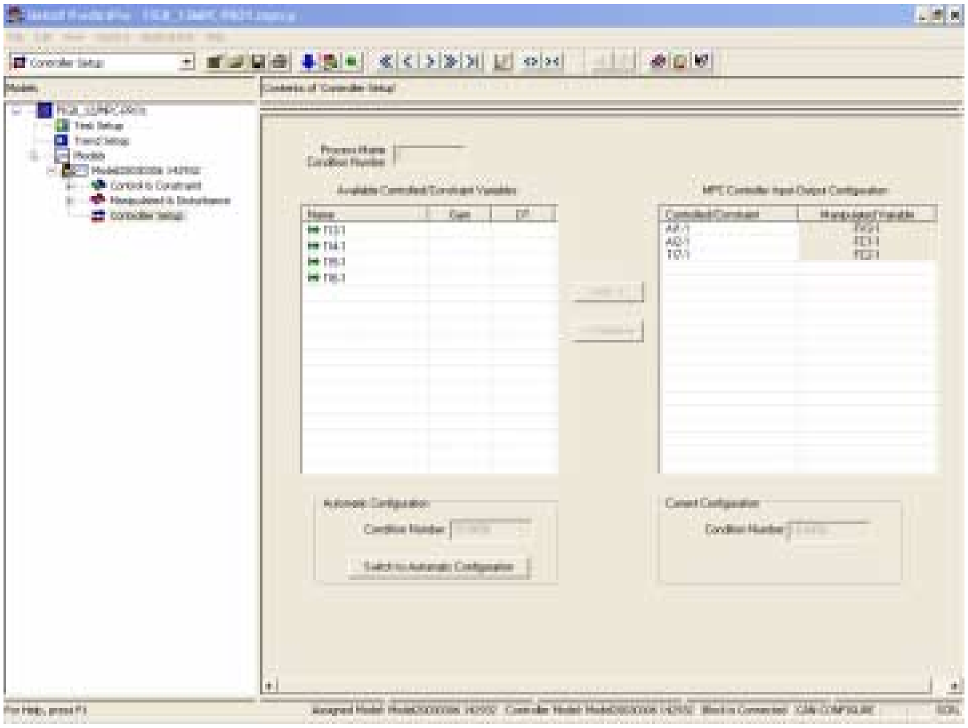
First Order Time Const: 45.00

Squared Error:0.00

The Control or Constraint parameter that best reflects changes in the Manipulate parameter are automatically selected for use in the controller generation. The Control or Constraint parameters that have been selected are represented in different colors in the step response for each Manipulate parameter. For example, the selection for the Top Draw Flow is shown in the following figure.



The selection of the parameters used in control is based on the parameters' impact on the condition number associated with the control matrix that is generated. Also, the deadtime and gain associated with a parameter should be considered in this selection. Preference is given to Control parameters if the performance is nearly the same as that given for Constraint parameters. You can examine the Control and Constraint parameters selected for the control generation by selecting Controller Setup in the left pane. Choose Option | Expert to see this option. For this example, the information displayed is shown in the following figure.

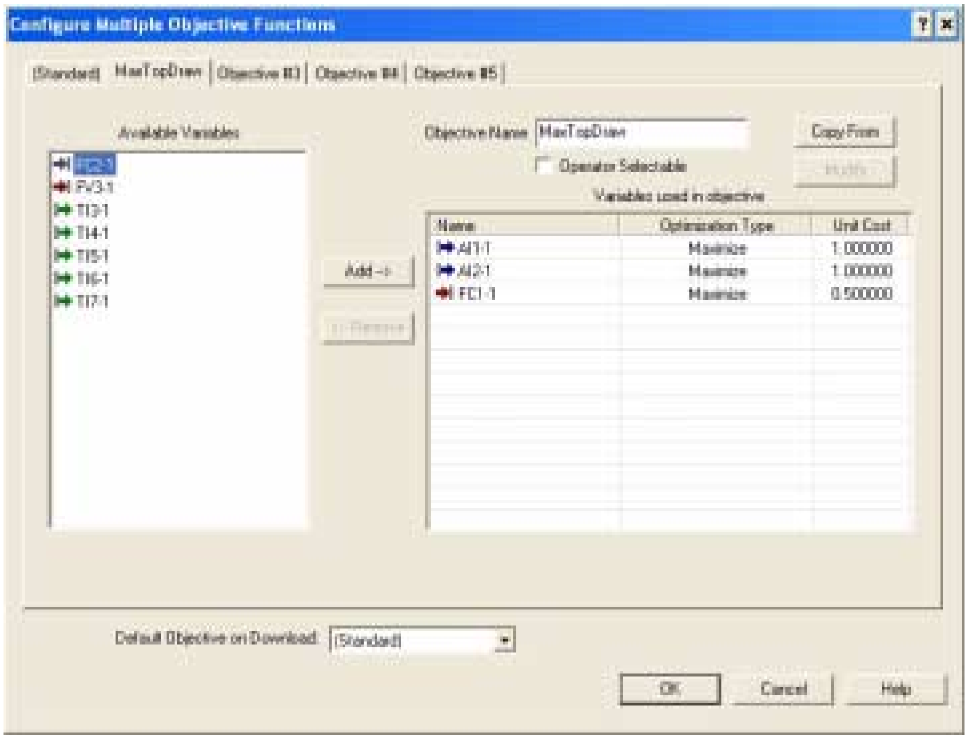


Name	Gain	DP
TI3-1		
TI4-1		
TI5-1		
TI6-1		

Controlled/Constraint	Manipulated Variable
AI1-1	FX2-1
AI2-1	FC1-1
TI2-1	FC2-1

For this example, the automatic choice of parameters gives a condition number of 18. If the Control and Constraint parameters are highly co-linear, that is reflecting the same information, it is possible that Control and Constraint parameters cannot be selected for some of the Manipulate parameters. Also, under certain conditions, the user can choose to force this condition by removing a parameter in the selection list as shown above – if this leads to a lower condition number. Generally, better control will be provided by a lower condition number.

The Standard objective function provided as the default for MPCPro can be used without modification for many applications. This default objective function is designed to provide control action that maintains the Control parameters at setpoint within the operating constraints. When there are extra degrees of freedom, for example, the number of manipulate parameters exceed the number of Controlled parameters as in this example, the Manipulated parameters will be minimized while maintaining the Control parameters at setpoint. When it is predicted that a Constraint parameter will violate one of its limits, the working setpoint of the Control parameters can be automatically modified within their specified range. If there is a conflict in satisfying the constraints and operating range, then the Control or Constraint parameters that are at their limit and are of lowest priority will be given up so that the remaining Control and Constraint parameters can be maintained within limits. For this application, the Standard objective function can be used. However, under certain market conditions, it can be desirable to maximize the top draw while maintaining the composition. Thus, a second objective function can be defined as shown in the following figure.



Configure Multiple Objective Functions

(Standard) | MaxTopDraw | Objective #3 | Objective #4 | Objective #5

Objective Name: MaxTopDraw

□ Operator Selectable

Available Variables:

- FC2-1
- FV3-1
- TI3-1
- TI4-1
- TI5-1
- TI6-1
- TI7-1

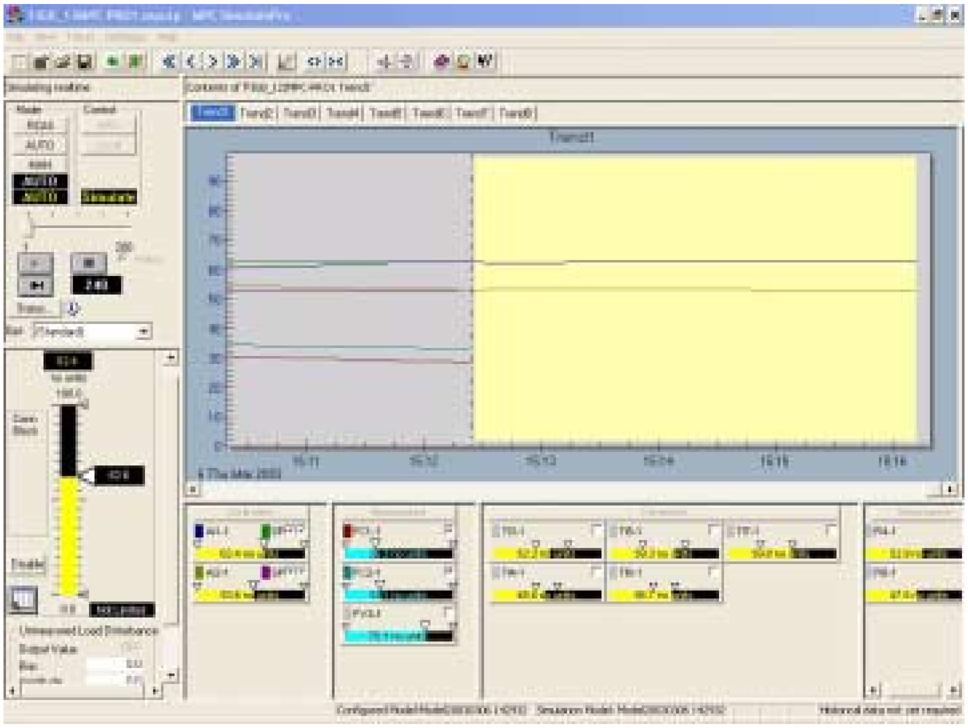
Variables used in objective

Name	Optimization Type	Unit Cost
AI1-1	Maximize	1.000000
AI2-1	Maximize	1.000000
FC1-1	Maximize	0.500000

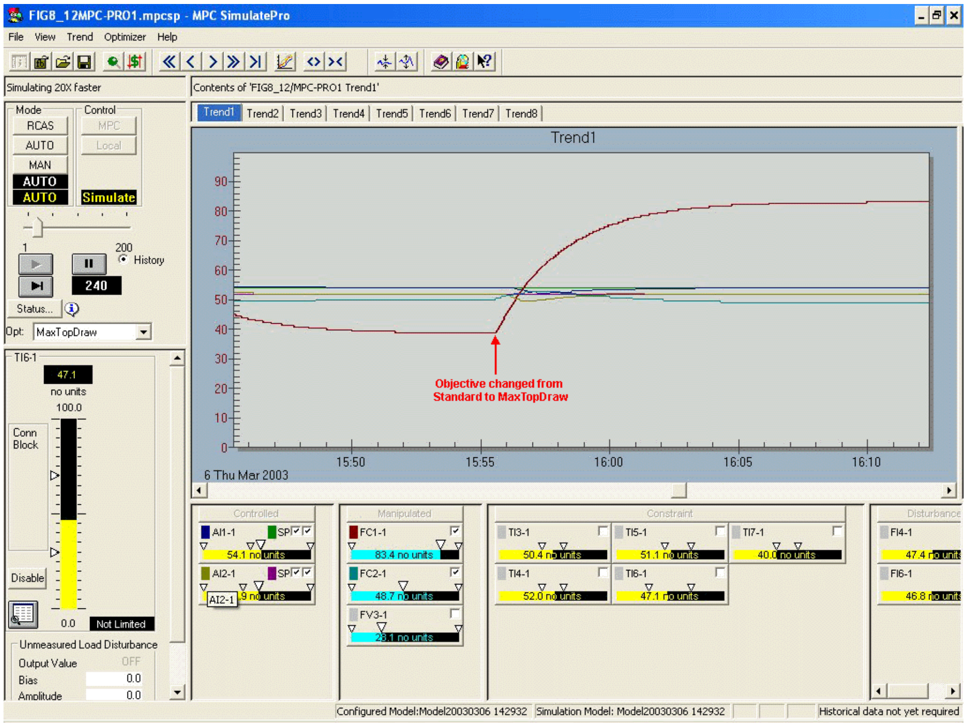
Default Objective on Download: (Standard)

In this case, the dollar profit associated with a percent change in the top and side end point composition and the top draw flow are not known or can change daily. Thus, the unit cost that is provided is used to indicate the relative value of maintaining composition versus maximizing top draw. For this objective, an objective name of MaxTopDraw is defined. Click the Operator Selectable checkbox for this objective to be available to the operator. Once any objectives have been defined, it is important to download the module to use these new objective functions in simulation and in on-line operation.

To test the column control off-line, select the model in the left pane and click the Simulate button. This command is available if you have selected Options | Expert. The following figure shows the Simulate interface for the example MPCPro block when MPCPro is in Automatic mode and the Standard objective function has been selected.



You can observe the control responses by changing the setpoint. You can also change the control objective to examine the impact on the process. For example, the impact of changing the objective from Standard to MaxTopDraw is reflected in the trend as shown in the following figure.



Simulating 20X faster

Contents of 'FIG8_12/MPC-PRO1 Trend1'

Opt: MaxTopDraw

Objective changed from Standard to MaxTopDraw

Unmeasured Load Disturbance

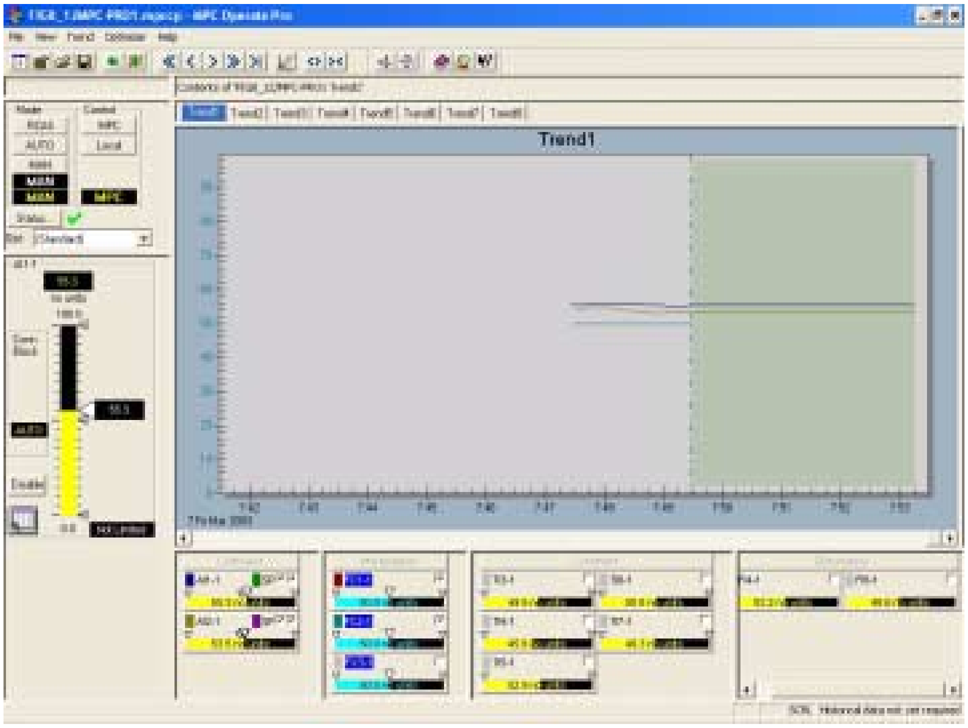
Output Value OFF

Bias 0.0

Amplitude 0.0

Configured Model:Model20030306 142932 | Simulation Model: Model20030306 142932 | Historical data not yet required

Once the MPCPro control has been verified using the simulation environment of PredictPro, the module containing the MPCPro block can be put into service. The predefined dynamos for PredictPro can be included in the Operator Interface. The following figure shows an MPCPro Operate view for this example.



Neural Network (NN) Function Block

Use DeltaV Neural Networks to create virtual sensors that continuously predict a process output based on measured process inputs. In addition, neural networks calculate and provide as a block output the future value of the output based on the current input values. Use neural networks where the process output is only available through analysis of lab samples or online analysis is not reliable.

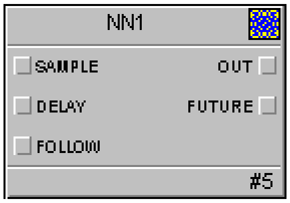
The NN function block is the basis of implementing neural networks in a DeltaV system. You define all parameters in the NN block using Control Studio. From Control Studio, you can launch the DeltaV Neural application to train the neural network.

If you are using a neural network to augment or replace the output of values determined by lab analysis, use the Lab Entry (LE) function block to provide the lab sample data to the NN block. When the neural network is providing backup for an online analyzer, use an Analog Input block to provide the measurement to the NN block. The neural network uses historical sample input data to train the neural network. After training, the neural network uses the sample input data to automatically correct the calculated output for changes in the process and unmeasured inputs. The ability of the NN block to adapt to changes reduces maintenance by minimizing the need to retrain the neural network to compensate for process changes.

Use the DeltaV Neural application to train and test neural networks. When a module containing an NN block is downloaded, all inputs and outputs are assigned to the Continuous Historian. The NN function block can predict the future output of a process given the current inputs, allowing for process dynamics. Refer to the DeltaV Neural topic for more information on neural network and training.

The operator runs the control through a DeltaV Operate picture constructed with the NN and LE (Lab Entry) dynamos.

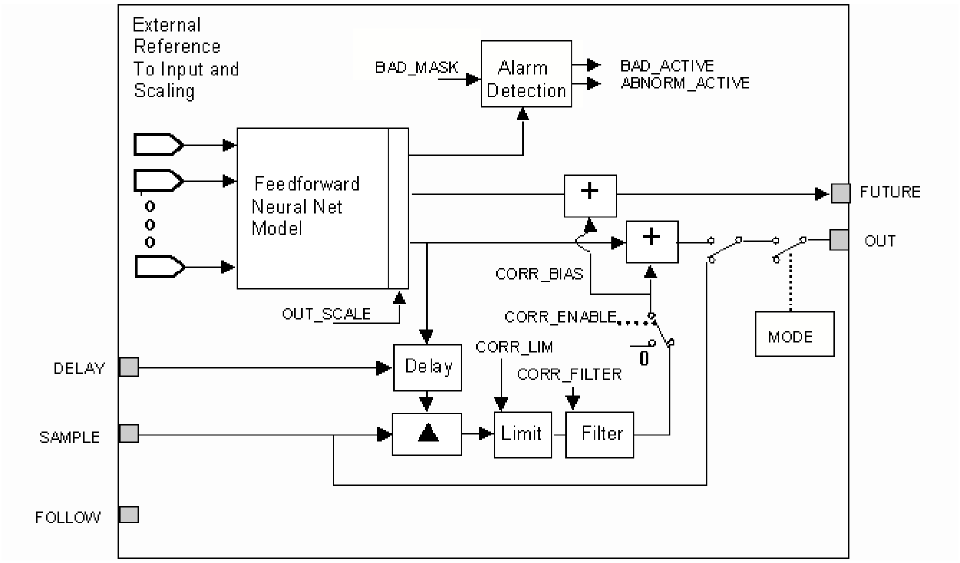
The NN function block supports modes and status handling.



Neural Network Function Block

Schematic Diagram - Neural Network Function Block

The following diagram shows the internal components of the Neural Network function block:



Neural Network Function Block Schematic Diagram

Configuration - Neural Network Function Block

Note Any change in the block configuration requires that you reconnect to the block from the Neural application for the change to be reflected in the block's behavior.

Implementing a neural network to model a process output requires that you create a module that contains a Neural Network function block. The tasks required are:

1 Create the module in Control Studio.

2 Specify the number of process inputs.

3 Assign the process output to model to the Sample input of the NN function block.

4 Connect the SAMPLE input to a Lab Entry or an Analog Input function block, depending on whether the parameter to be estimated is currently only available through offline lab analysis or is periodically measured using an online sampled analyzer.

5 Define process inputs to the model.

6 Specify the Historian Sampling Period.

Note Set the Historian Sampling Period only from the Properties dialog of the NN function block. Do ***not*** use the History Collection dialog.

7 Save the module and download it to the controller.

8 Verify that the transmitters that provide the inputs are working correctly.

9 Make sure the Continuous Historian is enabled, the area containing the module has been assigned to the historian and the Continuous Historian has been downloaded.

10 Start the Neural application and connect to the NN block.

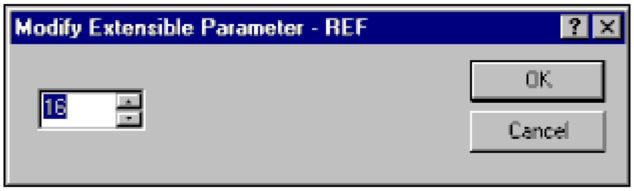
11 Select historical data and train the network.

12 Download the trained network to the controller.

Refer to the DeltaV Neural topic for more information on using the Neural application to train, test, and download your network model.

Defining Process Inputs

Because the number of inputs to a neural network can vary, the NN function block uses extensible parameters for process inputs. After you decide which process inputs may affect the process output you are modeling, right click the NN function block and select Extensible Parameters. The following dialog appears:



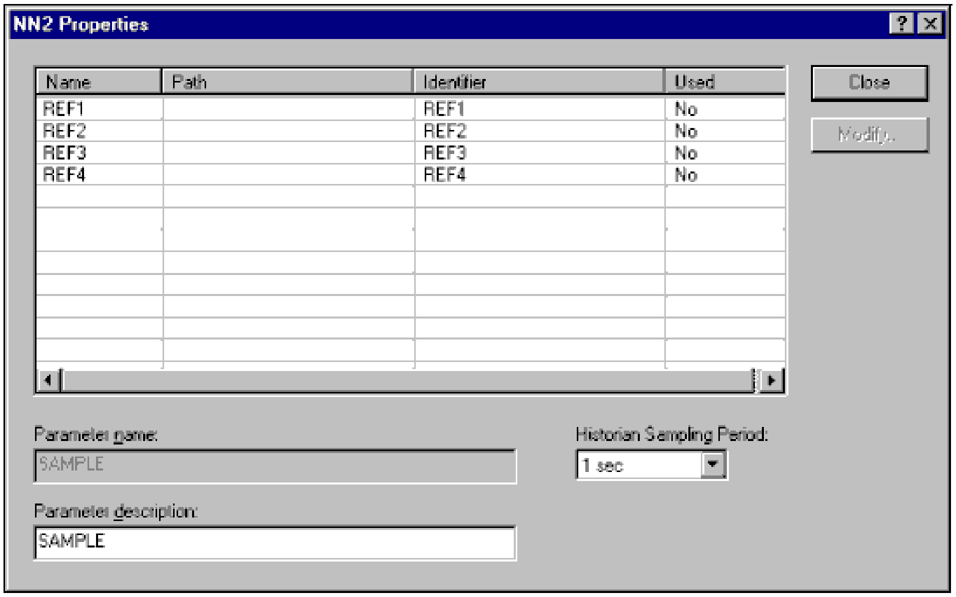
Modify Extensible Parameter - REF

16

Use the spin buttons to enter the number of extensible parameters you need for the number of process inputs you are using, and then click OK. You can configure as many as 20 variables as inputs to the Neural Network, but the model uses a maximum of 16 inputs. You (or the Neural application) eliminate variables when creating the model so that no more than 16 inputs remain.

Next, right click the NN block and select Properties. The Neural Network block Properties dialog appears with default process input parameter information. The dialog also contains the modeled parameter (the process output the network models) name and a default description. You can change the text in the Parameter description field if you desire.

Next, configure the Historian Sampling Period to define the rate at which the data is going to be stored in the DeltaV Historian. This should correspond to sampling period required by the fastest moving process input.



NN2 Properties

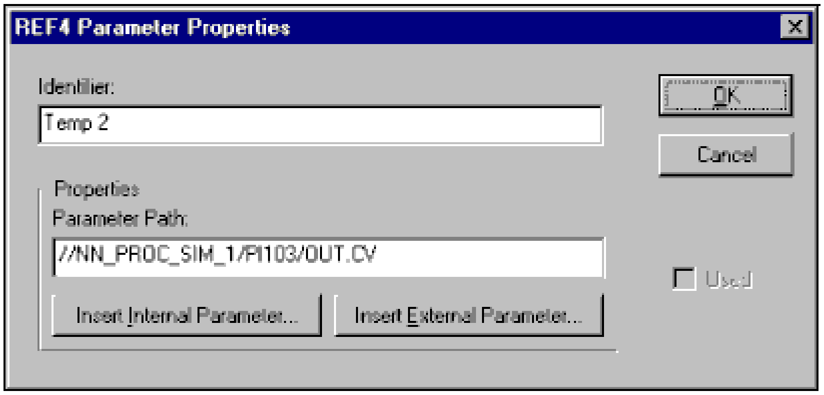
Name	Path	Identifier	Used
REF1		REF1	No
REF2		REF2	No
REF3		REF3	No
REF4		REF4	No

Parameter name: SAMPLE

Parameter description: SAMPLE

Historian Sampling Period: 1 sec

Use the Properties dialog to open the Parameter Properties dialog for all process inputs to configure the Neural Network's input parameters. Select a parameter and click the Modify button. The Parameter Properties dialog opens.



REF4 Parameter Properties

Identifier: Temp 2

Properties

Parameter Path: //NN_PROC_SIM_1/PI103/OUT.CV

☐ Used

Use the Parameter Properties dialog to specify a valid parameter path and a unique identifier for each process input. Note that the path and identifier must be unique for each process input parameter. Note that you cannot change the value in the Used field from either the Neural Network Properties or the parameter Properties dialog. That field indicates if the specific parameter is used in the neural network model assigned to the Neural Network block after training. Because the Neural Network is not trained yet, the Used fields default to No for all the input parameters.

Note Set the Historian Sampling Period only from the Properties dialog of the NN function block. Do ***not*** use the History Collection dialog.

After you make all the required changes to the module, you can save and download the module to the DeltaV controller. You must also download the Continuous Historian for data to be historically trended.

Block Execution - Neural Network Function Block

The value, status, and associated scaling of the configured inputs are reported unsolicited to the neural network (NN) block. In both Auto and Man modes, inputs selected during sensitivity analysis are used to calculate the process output. Also, the block calculates the future output of the process for the current input values. If the value of an input or output is outside the data range used to train the neural network, the value is limited to the data range. If correction is enabled (CORR_ENABLE) and the status of the sample is not limited, the block computes the model error by taking the difference between the current sample process output (SAMPLE) and the calculated Neural Network output delayed by the sample time (DELAY). After limiting the difference to +/- the correction limit (CORR_LIMIT), the block filters the model error to obtain the calculated model correction bias (CORR_BIAS). The block adds the calculated correction bias to the calculated process output (OUT) and future output (FUTURE). Note that the NN block retains OUT values upto a maximum delay of the Time to Steady State (TSS) value configured in the DeltaV Neural application. Therefore, a SAMPLE value entered with DELAY greater than TSS will not affect the CORR_BIAS value.

In Auto mode, the OUT parameter and FUTURE parameter of the block reflect the corrected process output and future value respectively. In Man mode, the block updates the FUTURE parameter only.

If one or more of the reference inputs is invalid, the NN block sets Readback Failed and Input Failure/Bad PV in BLOCK_ERR. The status of all of the inputs is set to Bad resulting in the OUT and FUTURE being set to Bad OOS (out of service). Also, in the trend parameter list the inputs have a value of 0.0 and in the trend view the inputs with invalid references have the value 0.

When one or more of the inputs used in the NN block has a status of Bad, the Input Failure/Bad PV is set in BLOCK_ERR.

If an input or output value is limited to the data range used in training the NN, the Other Error is set in BLOCK_ERR and the FUTURE and OUT have an Uncertain status. The value used in computing the NN prediction is limited to the training range. Typically, FUTURE precedes OUT in going to Uncertain because OUT uses the delayed inputs, which would have been within training range. When the value is back within training range, FUTURE again precedes OUT in returning to Good status. Your configuration of BAD_MASK determines whether these conditions cause BAD_ACTIVE or ABNORMAL_ACTIVE to be set.

If the FOLLOW input is active (1) and the mode is AUTO, then the OUT parameter value and status is forced to match the SAMPLE input value and status. When the module that contains the NN block is downloaded, the block is initialized using the current input values.

Modes - Neural Network Function Block

The NN function block supports three modes:

Out of Service (OOS)

Manual (Man)

Automatic (Auto)

For complete descriptions of the supported modes, refer to the Function Block Modes topic.

Status Handling - Neural Network Function Block

If the block is in Auto mode, the OUT and FUTURE status are set to Bad if any of the process inputs used in the NN block are Bad. When one or more inputs have a status Uncertain and the remaining inputs have Good status, the status of the outputs is set to Uncertain. Also, if inputs are Good and the input values are within the data range used to train the neural network, then the OUT status is set to Good. The OUT status is set to Uncertain if one or more of the input values is outside the data range used to train the neural network or if the OUT value is outside the training range.

In Manual mode, the status of FUTURE is determined the same as in Auto. However, the status of OUT is set to Good Constant status, unless its value is outside the training range, in which case it is Uncertain.

In Out of Service mode, the status of OUT and FUTURE is set to Bad.

If more than five inputs have values outside the training range, the corresponding prediction (OUT and/or FUTURE) has Bad status.

If an input has a Const or Uncertain status, OUT and FUTURE have Uncertain status.

Parameters - Neural Network Function Block

The following table lists the system parameters for the Neural Network function block:

Neural Network Function Block System Parameters

Parameter	Unit	Description
ABNORM_ACTIVE	None	The indication that a block error condition not selected in BAD_MASK (on the function block level) is True (Active).
BAD_ACTIVE	None	The indication that a block error condition selected in BAD_MASK (at the function block level) is True (Active).
BAD_MASK	None	The set of active error conditions that triggers a user-defined Bad condition. The user selects a subset of block error (BLOCK_ERR) conditions in the BAD_MASK parameter. When any of these conditions is True, the BAD_ACTIVE parameter becomes True. When any of the BLOCK_ERR conditions not included in BAD_MASK is True, ABNORM_ACTIVE becomes True.
BLOCK_ERR	None	The actual error status of the function block, set in the following manner: **Out of Service** – Block mode is out of service. **Readback Failed** – One or more of the inputs has an invalid reference path. **Input Failure/Bad PV** – One or more of the inputs used by the neural network have a Bad status or were configured with an invalid path. **Configuration Error –** The neural network module scan is greater than one second, and none of the inputs have a parameter path defined. **Other Error** – An input or output value exceeded the values in the data range used to train the neural network and is being limited to this range before being used.
CORR_BIAS	EU of OUT_SCALE	The model correction value added to the neural network output to determine OUT and FUTURE.
CORR_ENABLE	None	Sets the neural network model correction value used. If correction is enabled, the model correction value (CORR_BIAS) is calculated as the difference between the sampled value and the neural network output, after limiting and filtering. If correction is disabled, a value of zero (0) is used.
CORR_FILTER	Seconds	Time constant of the first order filter used to calculate the model correction factor. Filter saturation is prevented by calculating the filtered value only when CORR_BIAS is being computed and using the ratio between CORR_FILTER and block execution period.

Parameter	Unit	Description
CORR_LIM	EU of OUT_SCALE	Absolute value of the maximum difference between the sampled value and the neural network output. Limited to 10% of scale.
DELAY	Seconds	The time elapsed between sampling and lab analysis or sampled analyzer output being available to the NN block.
FOLLOW	None	In AUTO mode, the OUT parameter tracks the SAMPLE input value and status when FOLLOW is active (True). If FOLLOW is inactive (False), OUT is calculated based on the block algorithm.
FUTURE	EU of OUT_SCALE	The predicted value that the process output would become if the input values were to remain constant at their current value and status.
IN_DESC*n* where *n* is 1 through 16	None	User-specified unique description of IN*n*. These parameters appear only after the network has been trained.
IN*n* where *n* is 1 through 16	Determined by source or EU of PV_SCALE or EU of IN_SCALE	Input value and status. These parameters appear only after the network has been trained.
INSPECT_ACT	None	Indicates if Inspect is enabled and one or more of the limits for the block have been exceeded. The normal value is 0. This parameter is set by the Inspect application and is set to 1 if the following conditions are true: - The Write To Inspect Alarm context menu item was selected from Inspect for this block. - With the Current Hour filter selected, Inspect indicates that an abnormal condition exists for Mode, Control, Input, or Variability. (Note that an abnormal condition will only exist for Variability if both the Variability Index **and** the Standard Deviation have exceeded their defined limits.)
MODE	None	Parameter used to show and set the block operating state. MODE contains the actual, target, permitted, and normal modes.
OUT	EU of OUT_SCALE	The process output value calculated by the neural network based on process inputs.
OUT_SCALE	None	The high and low scale values, engineering units code, and number of digits to the right of the decimal point associated with OUT.
SAMPLE	EU of PV_SCALE	Sampled process output determined by lab analysis or by a sampled analyzer.

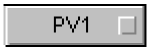
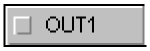
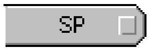
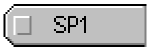
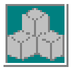
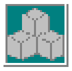
Parameter	Unit	Description
SAMPLE_DESC	None	Description of the sampled process output. This description appears in the parameter list in the Neural application.
STDEV	Percent of scale	The standard deviation of PV. For analog control blocks in AUTO, mean is assumed to be the SP. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_CAP	Percent of scale	The estimated capability standard deviation (measurement of short term variation). An estimate of the least standard deviation the process could achieve ideally. Refer to Loop Performance Calculations for more details on how this parameter is calculated.
STDEV_TIME	Seconds	The timeframe over which STDEV and STDEV_CAP are performed. The default value of zero is good for most processes where the scan rate is no more than approximately 10 times faster than the time to steady state. If the process is relatively much slower, it is recommended that you enter the approximate time it takes for the process to return to steady state after a change. This ensures that the STDEV and STDEV_CAP calculations accurately consider the actual time constant of the process.

Custom Blocks

You can create custom blocks that meet your specific needs. The custom block can be an embedded composite, a linked composite, or a module block. Use the Custom Block Assistant to create a custom function block.

Special Items Palette

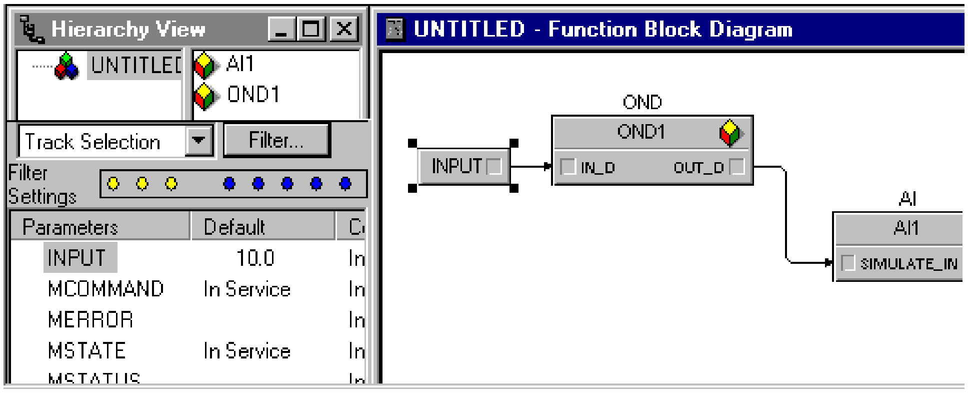
The Special Items palette helps you create algorithms.

 PV1 □	**Input Parameter** – A value from another function block, a sequential function chart (SFC), or a field device supplied to a module for use in execution.
 □ OUT1	**Output Parameter** – A value calculated or triggered by the module and sent to other function blocks, SFCs, or field devices.
 SP □	**Internal Read Parameter** – A parameter, normally used only within a module.
 □ SP1	**Internal Write Parameter** – A parameter, normally used only within a module.
	**Custom Block** – a tool that allows you to add embedded or linked composites, module blocks, or function blocks to your diagram.
	**Physical Block** – A tool that speeds up the building of modules for fieldbus devices.

Input Parameter

An input parameter appears automatically as an input connector when the object is included in a module or composite block. The input parameter provides an input connection that allows you to bring in signals from other modules or from the field.

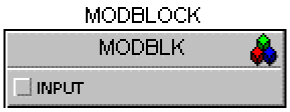
The following figure shows an example of an input parameter that has been added at the module level and how it looks on a function block diagram. Notice in the Parameter View that the INPUT parameter appears at the module level. If this module were used inside of another module on a function block diagram, you could directly wire a signal to that parameter.



Parameters	Default	C
INPUT	10.0	In
MCOMMAND	In Service	In
MERROR		In
MSTATE	In Service	In
MSTATUS		In

Example of Input Parameter Promoted to Module Level

Use the input parameter to make a parameter visible outside of its current level. If you had put this module with the input parameter into another function block diagram as a module block, you would see the input parameter as shown below:



Module Block View of Input Parameter

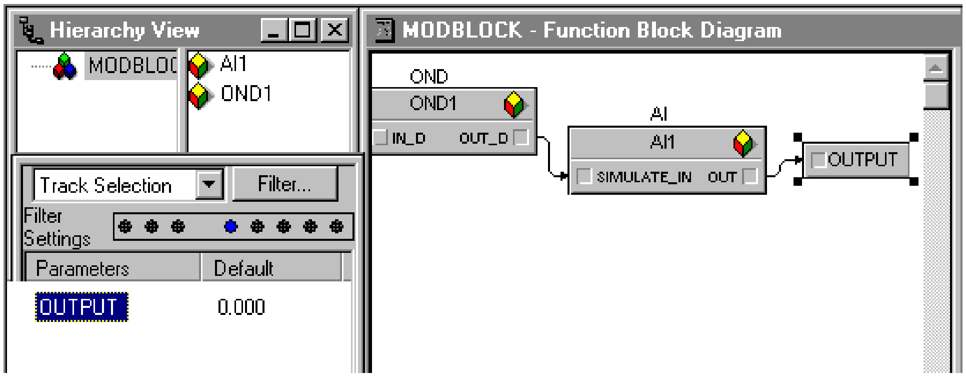
Note Do not reference an external parameter that references another external parameter. (You can configure this, but it will cause you problems at run time.)

Note In most cases, if the input parameter is used as an external reference to another module's parameter, the external reference will not work if the referenced parameter does have the CV field. There is one exception; you can reference Mode of a standard DeltaV function block. The Mode parameter does not have the CV field.

Output Parameter

An output parameter appears automatically as an output connector when the object is included in a module block or composite. The output parameter provides an output connection that allows you to send signals to other blocks, modules, displays, or to field devices.

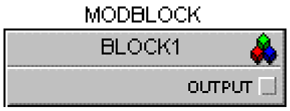
The following figure shows an example of an output parameter that has been added at the module level and how it looks on a function block diagram. Notice in the Parameter View that the OUTPUT parameter appears at the module level. If this module were used inside of another module on a function block diagram, you could directly wire a signal from that parameter.



Parameters	Default
OUTPUT	0.000

Example of Output Parameter Promoted to Module Level

Use the output parameter to make a parameter visible outside of its current level. If you had put this module with the output parameter into another function block diagram as a module block, you would see the output parameter as shown below:



Module Block View of Output Parameter

Note In most cases, if the output parameter is used as an external reference to another module's parameter, the external reference will not work if the referenced parameter does have the CV field. There is one exception; you can reference Mode of a standard DeltaV function block. The Mode parameter does not have the CV field.

Internal Read Parameter

An Internal Read Parameter provides a connector from which you can wire on a function block diagram. On an SFC, you can reference this parameter. The parameter connector **is not** seen outside of the block or module by default. You can use the Show Parameter function to make an Internal Read Parameter appear as an input connector on a module block or composite.

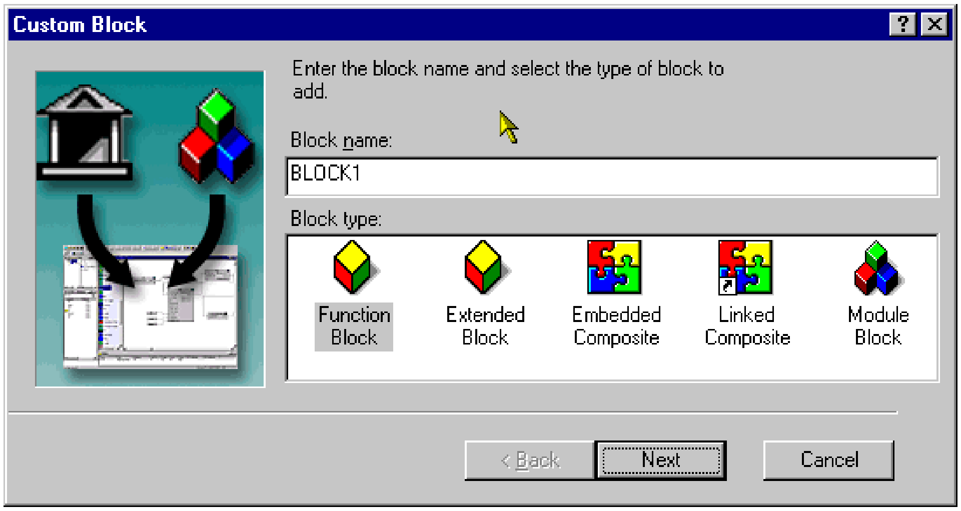
Internal Write Parameter

An Internal Write Parameter provides a connector to which you can wire on a function block diagram so that you can write a particular parameter. The parameter connector **is not** seen outside of the block or module by default. You can use the Show Parameter function to make an Internal Read parameter appear as an output connector on a module block or composite.

Note Make sure this parameter does not reference an external reference parameter from within this same module. (You can configure this, but it will cause problems at run time.)

Custom Block Assistant

The Custom Block Assistant helps you add different types of blocks to your diagram. You can add function blocks, extended blocks, embedded or linked composite blocks, or module blocks. The Block Assistant asks you what type of block you want to add and whether it is a new or existing block. You answer the questions and identify the block, and the block assistant adds it to your diagram.



Enter the block name and select the type of block to add.

Block name: BLOCK1

Block type: Function Block

Custom Block Wizard

Block Types

Function Block - A primitive or basic block that provides commonly used functions for your algorithm.

Extended Block - A DeltaV function block that has been defined to run in a fieldbus device.

Embedded Composite - A block that is not linked to the original object. An embedded composite contains one or more other blocks and parameters. If changes are made to the original object, this composite will be unaffected. Use this type when you want an independent, standalone copy of an existing composite.

Linked Composite - A block that is linked to a composite block or template. A linked block is a composite of one or more other blocks and parameters. If changes are made to the original composite block, the changes affect this block (unless the changes are made to a parameter that is no longer linked to the original template).

Module Block - A block that provides input/output connectors to another module from within the module you are editing. For example, when implementing cascade control, you could have an inner loop and an outer loop that execute at different rates. The module block you add here could control one loop, and the module you are editing could control the other. The scan rate is a property of a module.

Physical Block Assistant

You can use the Physical Block Assistant to speed up the configuration of modules for fieldbus devices. It allows you to convert and assign the function block to a block in the fieldbus device in one step rather than having to use a regular function block and then assign it to a fieldbus device.

Index