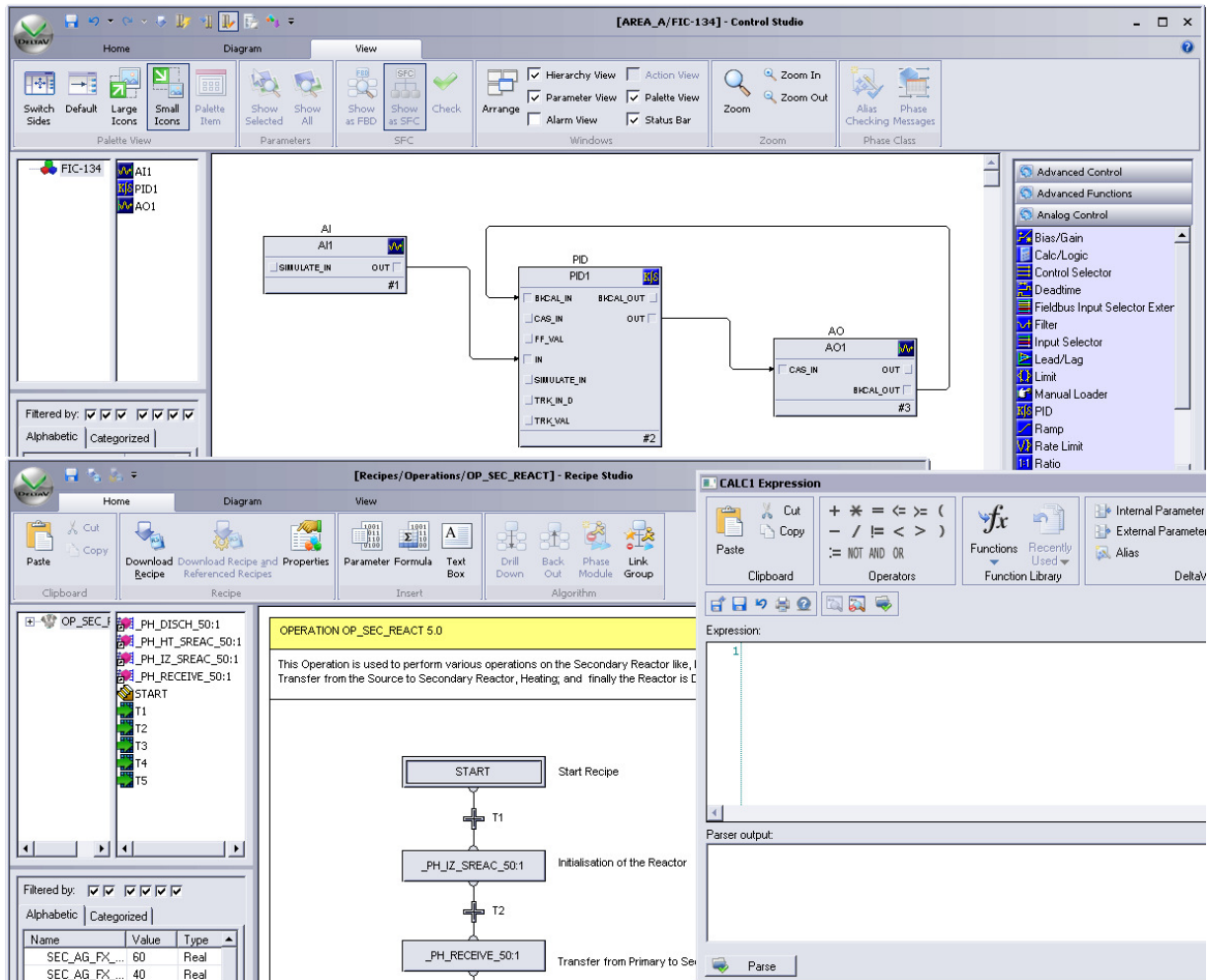


Monitor and Control Software



DeltaV™ allows you to use the right control language for the right control task.

- IEC61131-3 languages
- Built-in FOUNDATION fieldbus technology
- Control in the field, controller, or both
- Scalable software meets any needs
- Integrates multi-vendor real-time data into your DeltaV™ system
- Performs calculations on data from external sources
- Features protecting and hiding of control strategies
- Features class-based configuration
- Features advanced unit management
- Features advanced control



Introduction

Are you looking for a control system that allows you to use standard control languages and functions? Do you want to use different control languages that are appropriate for the control task within the same strategy? Do you want a control system that permits you to design where and how you perform monitoring and control, rather than forcing you to fit within the constraints of a fixed architecture? A DeltaV system gives you all this and more. Choose from [IEC61131-3](#) control languages and [FOUNDATION fieldbus](#) function blocks to design your monitoring and control strategies.

Need to consolidate your PLC, legacy systems, and gauging systems into a single operator interface? [OPC](#) may be used to tear down the barriers between your legacy control systems and your DeltaV system. The DeltaV system provides you with a real-time database where you can store third-party data for use anywhere within the DeltaV system. You can display, trend, alarm and even use it in control strategies. What's more, you choose the location that is best for integrating the data into the DeltaV system, the workstation or the controller.

This document details the monitoring and control functions supported by a DeltaV system. These functions can run on DeltaV Application Stations and controllers and many of the function blocks may be assigned to run within [FOUNDATION fieldbus](#) devices

Benefits

Uses IEC61131-3 control languages. A DeltaV system supports three languages, so you can always use the tools most appropriate for the job. Function Block Diagrams (FBD), Sequential Function Charts (SFC), and Structured Text (ST) are all available, making control strategy development both intuitive and easy.

Built-in FOUNDATION fieldbus functionality. These function blocks are used throughout the DeltaV system. Control strategies are developed, using these standards-based blocks that minimize development time and maximize the system's performance. Many functions (like override control, tracking, and state control) are built into these powerful blocks. Plus, with Fieldbus, you get decreased wiring costs and improved instrument diagnostics.

Control in the field, controller, or both. Design control strategies to meet your application requirements. Then decide where each block executes: in the controller, in the field devices, or both.

Scalable software meets any needs. As your needs grow, your DeltaV system is ready to expand with you. Additional capacity and even additional control functionality can be added online while you continue to control your process. Your entire configuration remains

intact; you simply add to existing control strategies with the newly available functions or configure new control modules to take advantage of higher capacity.

Integrates multi-vendor data. Bring data into your DeltaV system for control, display, and historical purposes. When executing in the controller, the monitoring software uses Classic I/O, HART, FOUNDATION fieldbus, PROFIBUS DP, AS-i bus, DeviceNet and serial interfaces to access data. Running in workstations, OPC technology provides Data Acquisition control software with fast connections and data transfers to and from any system.

Performs calculations on data from external sources. Now you can put the data from an existing system to work for you in the DeltaV system. All DeltaV applications can access and use the external data as if it were native DeltaV data. And because you can execute function blocks directly in the workstation, you can really put this external data to work for you.

Features protecting and hiding of control strategies. Access to library items like module templates and class definitions can be restricted by to enforce project standards. The internal structure of modules can be hidden to protect intellectual property.

Features class-based configuration. Build module class definitions for commonly used control (e.g. motors, valves) and equipment (e.g. valve header) modules. All modules that are created from a definition inherit changes that are made to the definition.

Features advanced unit management (AUM). Use powerful DeltaV unit modules and aliasing to write generic logic in control and equipment modules. This will simplify configuration management where there are similar equipment units or trains. Refer to the Advanced Unit Management product data sheet for further details.

Features advanced control. Simply run the tuning wizard and your loop is optimized. For more demanding process control requirements, you can use Model Predictive Control or DeltaV Neural. Refer to the Advanced Control product data sheets for details.

Product Description

DeltaV Monitoring and Control Software uses standard IEC61131-3 control languages, as well as FOUNDATION fieldbus function blocks to support strategy development. These strategies may be graphically assembled and modified using standard drag-and-drop techniques. Strategy development is visually intuitive, making it easy for first-time users to quickly become productive. Context-sensitive, online help is available for all functions. The IEC61131-3 control languages include Function Block Diagramming (FBD), Sequential Function Charts (SFC), and Structured Text (ST). Typical uses of each language are listed on page 4.

Control Language Typical Usages
<p style="text-align: center;">Function Block Diagram (FBD)</p> <p>Monitoring and alarming Continuous calculations Analog control (pressure, temperature, flow) Motor and block valve control Totalizers</p>
<p style="text-align: center;">Sequential Function Chart (SFC)</p> <p>Charging systems Startup/shutdown control Batch sequences (fill, mix, heat, dump)</p>
<p style="text-align: center;">Structured Text (ST)</p> <p>Advanced math functions Complex calculations Interlock condition detection If-then-else decisions Looping Bit manipulations</p>

Function Block Diagrams. Your DeltaV system uses Function Block Diagrams to implement continuously executing calculations, process monitoring, and control strategies. Graphical “wires” are used to connect the different blocks within a diagram. Each wire transfers one or more pieces of data. All communications throughout the DeltaV system are handled automatically. DeltaV function blocks are implemented using the structure specified by the FOUNDATION fieldbus standard, but they are also enhanced to provide greater flexibility in control strategy design. FOUNDATION fieldbus -compliant Function Blocks enable you to take advantage of control in the field. You design control strategies that best meet your control application requirements. Then, decide whether the function blocks controlling the process run in the controller, field devices, or both.

Sequential Function Charts. This language enables you to configure operator-independent time-variant actions. SFCs are best at controlling strategies with multiple states. They can be used for sequencing and simple batch applications.

An SFC comprises a series of steps and transitions. Each step contains a set of actions that affect the process. Transitions determine when processing moves on to the next step(s). Both single-stream and parallel execution of logic is supported within SFCs.













Structured Text. With structured text you can write sophisticated calculations using a wide array of algebraic and trigonometric functions and operators. In addition, you can build complex logic expressions using conditional and iterative structures.

Data Acquisition. Display, trend, alarm, and use third-party data as if it were native DeltaV data. Now you can put the data from an existing system to work for you in a DeltaV system. All DeltaV applications can access and use the external data as if it were native DeltaV data. The data can be used on any DeltaV workstation. It can also be integrated into control strategies, consolidated into reports, historized, and much more. Whether your application uses the workstation’s OPC server or the controller’s I/O subsystem—once accessed, the real-time database makes the data immediately available throughout the DeltaV system. In runtime, the *exception reporting* mechanism moves the data to the appropriate system application. The *global configuration database* makes management and use of this data easy.







Section 1: Function Blocks

Math Blocks	
	<p>Absolute* Provides the absolute value of an integer or floating point input</p>
	<p>Add* Sums the values of two to sixteen inputs and generates an output value.</p>
	<p>Arithmetic* Provides standard algorithms for : 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</p>
	<p>Comparator* Compares two values for the conditions 'greater than', 'less than', 'equal', 'not equal', and generates an output for each comparison (GT, LT, EQ, NEQ). It also checks whether the primary input is within the range defined by two inputs.</p>
	<p>Divide* Divides one input value by another input value and generates an output value.</p>
	<p>Integrator* Integrates one or 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.</p>
	<p>Multiply* Multiplies two to sixteen inputs and generates an output value.</p>
	<p>Subtract* Subtracts one input value from another input value and generates an output value.</p>














**Allowed to run in a workstation-based controller*



Input/Output Blocks	
	<p>Alarm Detection*</p> <p>Creates a set of standard alarms (HI_HI, HI, LO, LO_LO) on any system parameter. AI and PID blocks have this function built-in.</p>
	<p>Analog Input*</p> <p>Accesses a single analog measurement value and status from an I/O channel. The input value can be a transmitter's 4-20mA signal, the digitally communicated FOUNDATION fieldbus block, or the digitally communicated primary or non-primary variable from a HART transmitter.</p>
	<p>Analog Output*</p> <p>Assigns an analog output value to a field device through a specified I/O channel or Fieldbus device.</p>
	<p>Discrete Input*</p> <p>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.</p>
	<p>Discrete Output*</p> <p>Takes a binary setpoint and writes it to a specified O/O channel to produce an output signal.</p>
	<p>Fieldbus Multiplexed Discrete Input</p> <p>Accesses discrete input values from two-state field devices and makes the processed physical inputs available through eight discrete output parameters.</p>
	<p>Fieldbus Multiplexed Analog Input</p> <p>Used to make up to 8 input parameters available to the Fieldbus I/O subsystem.</p>
	<p>H1 Carrier Multiple Discrete Input</p> <p>Accesses a DI card on the H1 carrier device and makes the individual channels available as in an 8-bit integer output.</p>
	<p>H1 Carrier Multiple Discrete Output</p> <p>Takes an 8-bit integer and writes it to the individual channels of a DO card on the H1 carrier device.</p>
	<p>Fieldbus Multiplexed Analog Input</p> <p>Used to connect to higher density Fieldbus transmitters.</p>
	<p>Fieldbus Multiplexed Analog Output</p> <p>Used to connect up to 8 analog output signals for Fieldbus devices supporting this function block.</p>
	<p>Pulse Input*</p> <p>Converts the rate parameter on the Pulse Input card to a value in engineering units.</p>

**Allowed to run in a workstation-based controller*
















Timer and Counter Blocks	
	<p>Counter*</p> <p>Generates a discrete output value of True (1) when the count reaches a specified trip value. The block functions as an increment or decrement counter.</p>
	<p>Off-delay Timer*</p> <p>Delays the transfer of a False (0) discrete input value to the output by a specified time period.</p>
	<p>On-delay Timer*</p> <p>Delays the transfer of a True (1) discrete input value to the output by a specified time period.</p>
	<p>Retentive Timer*</p> <p>Generates a True (1) discrete output after the input has been True for a specified time period. When the reset input is set True, the elapsed time and the output value are reset.</p>
	<p>Timed Pulse*</p> <p>Generates a True (1) discrete output for 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; it returns to its original False value only when the output is True longer than the specified time duration.</p>
	<p>Date Time Event*</p> <p>Generates a pulse at a preset date and time or at preset time intervals</p>





**Allowed to run in a workstation-based controller*

Logic Blocks	
	<p>Action*</p> <p>Evaluates a single structured text expression when the input value is True. Mathematical functions, logical operators, and constants may be used in the expression.</p>
	<p>AND*</p> <p>Generates a discrete output value based on the logical AND of two to sixteen discrete inputs.</p>
	<p>Bi-directional Edge Trigger*</p> <p>Generates a True (1) discrete pulse output when the discrete input makes a positive (0-to-1, False-to-True) or a negative (1-to-0, 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.</p>
	<p>Boolean Fan Input*</p> <p>Generates a discrete output based on the weighted binary sum, binary coded decimal (BCD) representation, transition state, or logical OR of 1 to 16 discrete inputs.</p>
	<p>Boolean Fan Output*</p> <p>Decodes a binary weighted floating point input to individual bits and generates a discrete output value for each bit (as many as sixteen outputs).</p>
	<p>Condition*</p> <p>Evaluates a single structured text expression and generates a discrete output value when the expression is evaluated True (1) for longer than a specified time period.</p>
	<p>Device Control</p> <p>Provides setpoint control for multi-state discrete devices, such as motors, pumps, and block valves.</p>
	<p>Multiplexer*</p> <p>Selects one input value from up to 16 input values and places it at the output.</p>
	<p>Negative Edge Trigger*</p> <p>Generates a True (1) discrete pulse output when the discrete input makes a negative (1-to-0, True-to-False) transition since the last execution of the block. If there has been no transition (or a positive transition), the discrete output of the block is False (0).</p>
	<p>NOT*</p> <p>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.</p>
	<p>OR*</p> <p>Generates a discrete output value based on the logical OR of 2 to 16 discrete inputs. When one or more of the inputs is True (1), the output is set True (1).</p>
	<p>Positive Edge Trigger*</p> <p>Generates a True (1) discrete pulse output when the discrete input makes a positive (0-to-1, False-to-True) transition since the last execution of the block. If there has been no transition (or a negative transition), the discrete output of the block is False (0).</p>
	<p>Reset/Set Flip-flop*</p> <p>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.</p>












Logic Blocks	
	<p>Set/Reset Flip-flop*</p> <p>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.</p>
	<p>Transfer*</p> <p>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.</p>

**Allowed to run in a workstation-based controller*






Analog Control Blocks	
	<p>Bias/Gain</p> <p>Provides adjustable gain capability by computing an output value from a bias setpoint, an input, and a gain value.</p>
	<p>Calc*</p> <p>Allows you to specify an expression that determines the block's output. Mathematical functions, logical operators, constants, parameter references, and I/O reference values may be used in the expression. Expressions are written using IEC 61131-3 Structured Text. See section 3 for Structured Text details.</p>
	<p>Control Selector</p> <p>Selects one of three control signals to perform override control to a PID function block.</p>
	<p>Dead time*</p> <p>Introduces a pure time delay in the value and status used in a signal path between two function blocks.</p>
	<p>Fieldbus Input Selector Extended</p> <p>Selects the first good, minimum, maximum, middle, or average of up to eight input signals. Supports operator-selected, mode, and signal disable.</p>
	<p>Filter*</p> <p>Applies an equation to filter changes in the input signal and generates a smooth output signal.</p>
	<p>Input Selector*</p> <p>Selects the first good, minimum, maximum, middle, or average of up to four input signals. Supports operator-selected, mode, and signal disable.</p>
	<p>Lead/Lag*</p> <p>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.</p>
	<p>Limit*</p> <p>Limits an input value between two reference values.</p>
	<p>Manual Loader</p> <p>Allows the block output to be set by an operator.</p>
	<p>PID</p> <p>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.</p>
	<p>Ramp*</p> <p>Creates a ramping output signal to increase or decrease a variable toward a specified target value at a defined rate.</p>
	<p>Rate Limit*</p> <p>Limits the rate of change of the output value to specified limits. Also provides the derivative of the input signal.</p>
	<p>Ratio</p> <p>Applies an adjustable ratio setpoint to achieve a desired input/output relationship.</p>
	<p>Scaler*</p> <p>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.</p>

Analog Control Blocks	
	<p>Signal Characterizer*</p> <p>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.</p>
	<p>Signal Generator*</p> <p>Produces an output signal used to simulate a process signal.</p>
	<p>Signal Selector*</p> <p>Selects the maximum, minimum, or average of as many as sixteen input values and places it at the output.</p>
	<p>Splitter</p> <p>Takes a single input and calculates two outputs based on specified coordinate values.</p>










**Allowed to run in a workstation-based controller*

Advanced Control Blocks	
	<p>Fuzzy Logic</p> <p>Easy to use alternative to PID control. Useful in non-linear and noisy processes. Also very suited to handling unmeasured disturbances.</p>
	<p>Inspect*</p> <p>Monitors plant area control performance statistics for viewing, trending and history collection. Also provides for programmatic enable/disable of performance monitoring based on equipment or unit state.</p>
 	<p>Model Predictive Control* (basic and professional)</p> <p>Model Predictive constrained multi-variable controller for processes with, interacting variables, difficult dynamics, multiple constraints, and throughput optimization.</p> <p>Related Blocks</p> <p> MPC Process Simulator*</p> <p>Allows the process to be simulated to verify the model.</p> <p> MPC Input Reference*</p> <p>Easily connect input parameters (controlled variables, disturbance variables, and constraint variables) from multiple loops in a single MPC strategy. Useful for implementing MPC in a supervisory fashion.</p> <p> MPC Output Reference*</p> <p>Easily connect output parameters (manipulated variables) from multiple loops in a single MPC strategy. Useful for implementing MPC in a supervisory fashion.</p> <p> MPC Analyzer template*</p> <p>Used to pre-process sampled analyzer measurements before feeding them into the MPCPro block.</p>
	<p>Neural Network*</p> <p>Build virtual sensors for on-line qualitative property estimation. Use anywhere samples are taken for analysis, or in place of or back-up to continuous analyzers.</p> <p>Related Blocks</p> <p> Lab Entry*</p> <p>Allows operator input of offline laboratory analysis results to be used in training the neural network. Sample time and test results are easily logged right from the operator interface.</p>
	<p>Diagnostic*</p> <p>The Diagnostic function block provides a method to monitor device alerts from non-fieldbus assets.</p>









**Allowed to run in a workstation-based controller*

Advanced Functions Blocks	
	<p>Analog Voter</p> <p>Monitors a number of analog input values and determines if there are enough votes to trip. If a configured number of the inputs vote to trip, the block trips, and the output of the block is set to zero (0).</p>
	<p>Discrete Voter</p> <p>Monitors a number of discrete input values and determines if there are enough votes to trip. If a configured number of the inputs vote to trip, the block trips, and the output of the block is set to zero (0).</p>
	<p>Cause and Effect Matrix</p> <p>Incorporates interlock and permissive logic for one or more final control elements. Up to 16 Causes (inputs) and up to 16 Effects (outputs) can be configured using the function block's matrix.</p>
	<p>State Transition</p> <p>Implements a user-defined state machine. The state machine describes the possible states, and the transitions between those states, that can occur.</p>
	<p>Step Sequencer</p> <p>Associates system states with actions to drive outputs based on the current state.</p>

**Allowed to run in a workstation-based controller*









Energy and Metering Blocks	
	<p>Isentropic Expansion*</p> <p>Calculates the final enthalpy for isentropic expansion of steam to a given pressure for 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 (% moisture) and steam temperature are also calculated.</p>
	<p>Saturated Steam Properties*</p> <p>Calculates steam enthalpy, entropy, specific volume and pressure for saturation conditions specified by a given temperature.</p>
	<p>Saturated Temperature*</p> <p>Calculates steam temperature at saturation give steam pressure.</p>
	<p>Steam Density Ratio*</p> <p>Calculates the square root of the ratio of steam density to the density of steam at the calibration conditions of the flow meter.</p>
	<p>Steam Properties*</p> <p>Calculates steam enthalpy, entropy and specific volume for a give gauge pressure and a given temperature.</p>
	<p>Water Enthalpy*</p> <p>Calculates enthalpy of water for a specified temperature.</p>
	<p>Water Entropy*</p> <p>Calculates entropy of water for a specified temperature.</p>
	<p>Flow Metering in SI units*</p> <p>Calculates mass flow, volumetric flow and energy flow for orifice and turbine meters.</p>
	<p>Flow Metering in US units*</p> <p>Calculates mass flow, volumetric flow and energy flow for orifice and turbine meters.</p>

**Allowed to run in a workstation-based controller*

Special Items	
	<p>Custom Block Tool* Adds various block types</p> <p style="text-align: center;">  Composite Block* A grouping of individual function blocks. Composites can be of the Linked or Embedded type. </p> <p style="text-align: center;">  Module Block* A block on the diagram that represents another module within the system. It allows direct connection of parameters between this module and the referenced module. </p>
	<p>Physical Block Tool Used to add Foundation fieldbus blocks directly to the diagram instead of adding a standard I/O block and then converting it.</p>
	<p>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.</p>
	<p>Internal Read Parameter* A value used within a module that supplies an output for connection to I/O or other module parameters.</p>
	<p>Internal Write Parameter* A value used within a module that supplies an input for connection to I/O or other module parameters.</p>
	<p>Output Parameter* A value calculated or triggered by the module and sent to other diagrams or field devices.</p>

**Allowed to run in a workstation-based controller*

Section 2: Sequence Function Chart Tools

SFC Tools	
	<p>Input Parameter</p> <p>A value from another function block, a sequential function chart (SFC), or a field device supplied to a module for use in execution. Automatically available when the Module Block is used on another module's diagram.</p>
	<p>Internal Read Parameter</p> <p>A value used within a module that supplies an output for connection to I/O or other module parameters.</p>
	<p>Internal Write Parameter</p> <p>A value used within a module that supplies an input for connection to I/O or other module parameters.</p>
	<p>Output Parameter</p> <p>A value calculated or triggered by the module and sent to other function blocks, SFC's, or field devices. Automatically available when the Module Block is used on another module's diagram.</p>
	<p>Step</p> <p>An element of an SFC that contains a set of actions. The actions define the rules and the behavior of the step. The step manages the execution of the actions they contain.</p>
	<p>Termination</p> <p>A transition that specifies a condition that represents a logical end to a sequence.</p>
	<p>Transition</p> <p>An element of a SFC that allows the sequence to proceed from one step to the next if a transition condition is met. A transition condition is the result of evaluating a Boolean expression.</p>
	<p>Sequence</p> <p>A number of steps and transitions linked together.</p>

Section 3: Structured Text Instructions

The DeltaV system uses IEC 61131-3 Structured Text to create custom functions. The language uses operators, functions, statements and operands in expressions. The Action, Condition, Calc blocks and SFCs use expressions. Any system parameter may be referenced by these expressions.

Operators

Operator	Description
-	Unary minus
+	Unary plus
!, NOT	Logical NOT
~	Bitwise NOT
OR	Logical OR
XOR	Logical Exclusive OR
AND	Logical AND
	Bitwise OR
^	Bitwise XOR
&	Bitwise AND
=	Equality
!= , <> , ~=	Inequality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
+	Add
+	String concatenation
-	Subtract
*	Multiply
/	Divide
MOD , %	Modulus (remainder)
**	Raise to power
?:	Conditional evaluation

Functions

Math	Description
ABS (x)	Absolute value of x
EXP	Exponential of x
EXPT (x, y)	Raise x to the power of y
FRACT (x)	Fractional part of a number
LN (x)	Natural logarithm of x
LOG (x)	Log base 10 of x
LOG2 (x, y)	Log base y of x
MAX (x, y)	Maximum of x and y
MIN (x, y)	Minimum of x and y
ROUND (x)	Round x to the nearest float
SQRT (x)	Square root of x
TRUNC (x)	Floor function
Trig	Description
ACOS (x)	Arc cosine of x
ASIN	Arc sine of x
ATAN	Arc tangent of x
COS	Cosine of x
SIN(x)	Sine of x
TAN (x)	Tangent of x
Bitwise	Description
ROL(x,y) / ROT	Rotate x to the left by y positions
ROTL16(i,n)	Rotate i to the left by n bits for a 16 bit integer
ROR(x,y) / ROT	Rotate x to the right by y positions
ROTR16(i,n)	Rotate i to the right by n bits for a 16 bit integer
SHL(x,y)	Shift x left by y positions, no carry
SHR(x,y)	Shift x right by y positions, no carry
ASR16(i,n)	Arithmetic shift right n bits by l positions for a 16 bit integer
STBT(i,b,n)	Set bit n in integer i to boolean value b
Time	Description
time_to_str	converts a time value to a time string
str_to_time	converts a time string to a time value

Miscellaneous.	Description
EUP(e,p0,p100)	Conversion from engineering units to percent
PEU(p,p0,p100)	Conversion from percent to engineering units
Option Explicit	Forces all local variables to require explicit declaration
Logevent(x)	Logs the event string specified
Selstr(n,s1,s2,s3,s4,s5)	Returns the nth string.
Sysstat(x)	Download, Power fail, and switchover event flags
Basepath(path_string)	Returns the base portion of a path string.

Statements

Statement	Description
;	End of statement separator
:=	Assignment
If...Then... Else...End_if	Conditional execution
While...Do... End_While	Looping function
Exit	Exits the innermost While loop
Var...End_var	Declare variables
Option Explicit	Force variables to be declared
REM	Single line comment
(*...*)	Comment (multi-line)

Operands

- Named set constants in the form "SetName:ValueName."
- Inputs are floating-point-with-status data types that may be wired to function block parameter connectors or used in expressions.

- Outputs are floating-point-with-status data types that may be wired from function block parameter connectors or written from action expressions.
- Input device signal tags (DSTs) are references to signals and also contain signal status data.
- Constants- predefined, unchangeable values in expressions. These constants let you test values against DeltaV system values without having to know their internal representations. The DeltaV system supports the constants listed below:

Constant	Description
GOOD	Good status
BAD	Bad status
TRUE	Boolean True
FALSE	Boolean False
UNC	Uncertain status
LIMITED CONSTANT	Limited constant status
LIMITED HIGH	Limited High status
LIMITED LOW	Limited Low status
AUTO	Automatic mode
CAS	Cascade mode
IMAN	Initializing Manual mode
LO	Local Override mode
MAN	Manual mode
OS	Out of Service mode
RCAS	Remote Cascade mode
ROUT	Remote Out mode

Capacity

Your DeltaV system provides the exact monitoring and control functions your application requires. The software is also scalable in size. The system is sized according to the number of device signal tags (DSTs) and SCADA tags. A DST is a signal used as a measured variable or controlled output. A SCADA tag is a value brought into the DeltaV system for monitoring, display, alarming and history collection.

Classic I/O supports only one signal per device. HART, FOUNDATION fieldbus, PROFIBUS DP, ASi bus and serial communications can support multiple signals per device, but in most cases only 1 DST is counted for system sizing purposes. OPC communications can support multiple tags per server.

System Specifications	
Hardware	MX or MQ Controllers
Capacity SCADA tags Device signal tags	25000 per workstation 30000 per system
Data source	Classic I/O (AI, AO, DI, DO) HART (AI, AO) FOUNDATION fieldbus PROFIBUS DP, AS-i bus, DeviceNet, Serial, OPC

Ordering Information

Controller function licensing is based on the number and type of I/O connected. The license limits are enforced on a system-wide basis and not on a per controller basis. For details, refer to the Controller Licensing white paper.

License Type	Model Number	License Size (DSTs)
DI	CE3111Sxxxxx	xxxxx = 00025 to 30000
DO	CE3112Sxxxxx	xxxxx = 00025 to 30000
AI	CE3113Sxxxxx	xxxxx = 00025 to 30000
AO	CE3114Sxxxxx	xxxxx = 00025 to 30000
ProfessionalPLUS Station, for v10.3 and earlier systems	CE2101Sxxxxx	xxxxx = 00025 to 8000

Related Products

- **Redundancy.** Controller redundancy provides you with added reliability for critical applications.
- **DeltaV Neural.** Provide a continuous indication of lab analysis of difficult measurements.
- **DeltaV InSight.** An advanced process monitoring system that instantly identifies under-performing loops, provides comprehensive loop performance analysis tools, and allows easy tuning of control loops.
- **DeltaV PredictPro.** Obtain greater process throughput to maintain plant operation at measurable operating constraints.
- **DeltaV Simulate.** Development environment including simulation on a PC.
- **AMS Device Manager.** Provides predictive diagnostics and full asset management capabilities including calibration, documentation, and device configuration for HART, FOUNDATION fieldbus, Profibus DP, and WirelessHART devices.

To locate a sales office near you, visit our website at:

www.EmersonProcess.com

Or call us at:

India: +91 22 66620566

Asia Pacific: 65.6777.8211

Office Address:

Emerson Process Management (India) Pvt. Ltd.

Delphi B-Wing,

601-602, 6th Floor, Central Avenue,

Hiranandani Business Park,

Mumbai-400076

© Emerson Process Management 2016. All rights reserved. For Emerson Process Management trademarks and service marks, go to: <http://www.emersonprocess.com/home/news/resources/marks.pdf>.

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 warranties or guarantees, express 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.



DELTAV

www.DeltaV.com



EMERSON
Process Management

