G2 Diagnostic Assistant

Reference Manual

Version 5.1 Rev. 0

G2 Diagnostic Assistant Reference Manual, Version 5.1 Rev. 0
May 2007

Gensym Corporation
52 Second Avenue
Burlington, MA 01803 USA
Telephone: (781) 265-7100
Fax: (781) 265-7101

Part Number: DOC115-550

Contents Summary

Contents

Chapter 7 Time Series *(continued)*

Chapter 8 Statistical Process Control **189**

Chapter 23 Capabilities and Restrictions *(continued)*

Chapter 24 Connections **567**

Chapter 24 Connections *(continued)*

Chapter 25 Network Interfaces **577**

Chapter 26 Rule Terminals **597**

Preface

Describes this reference manual and the conventions that it uses.

About this Manual

GDA, the G2 Diagnostic Assistant, is an environment for developing and running intelligent operator applications. Its principal component is a graphical language that lets you express complex diagnostic procedures as a diagram of blocks, also called an Information Flow Diagram (IFD). These blocks are connected by paths that show how data flows through the diagram.

This manual documents each block that can appear in these diagrams.

This manual is strictly a reference and does not teach you how to use GDA. It assumes you are already familiar with G2 and GDA.

- See the *GDA User's Guide* for basic information on how to use GDA, and for information on how to customize GDA blocks and the GDA environment.
- See also the new *GDA API Reference* for a description of all the procedures that make up the API for the Gensym Diagram Language (GDL). You can use the API to modify and control diagrams programmatically.

In addition, the Educational Services Department at Gensym holds classes that are excellent ways to become familiar with this product.

Audience

This document is written for GDA application developers.

It is assumed that the reader is knowledgeable about G2, including how to write procedures and methods; and about GDA, including its class structure.

Organization

This manual is a menu-by-menu, palette-by-palette description of all the blocks in GDA. Each submenu has its own part, each palette in a submenu has its own chapter, and each block has its own section in that chapter.

Data Blocks

Part I documents the palettes in the Data Blocks submenu of the Palettes menu. These palettes contain blocks that manipulate data values. The part "Data Blocks" contains these chapters:

This chapter...	Documents blocks that...
Entry Points	Let you enter data into your GDA application. Entry Points let you get data from a G2 procedure, variable, parameter, or GSI.
Signal Generators	Let you simulate values and test your application.
Data Filters	Filter out noise and find trends in data.
Arithmetic	Perform arithmetic operations on data values: addition, subtraction, multiplication, and division.
Functions	Perform statistical operations (such as computing the average or median), perform linear algebraic operations (such as predicting future values for points on a line), and let you define your own function to apply to data values.
Data Control	Control how data flows through your diagram.

This chapter...	Documents blocks that...
Time Series	Perform operations on a history of values, such as averaging or integrating.
Statistical Process Control	Use statistical methods to measure a process's productivity and quality control.

Inference Blocks

Part II documents the palettes in the Inference Blocks submenu of the Palettes menu. These blocks create and manipulate inference (truth and fuzzy truth) values. The part "Inference Blocks" contains these chapters:

This chapter…	Documents blocks that…
Observations	Test data values to produce inference values.
Logic Gates	Combine inference values and control the flow of inference data.
Tabular Gates	Let you create logic tables.
Evidence Gates	Are specifically designed to work with fuzzy belief values to draw conclusions, convert belief values to data values, and test whether a belief value falls into a certain range.
Temporal Gates	Perform operations over a history of inference values, pass on time stamps, and analyze the timing of events.
Counters & Timers	Delay, time, and count inference values. They all have a digital readout to reflect their status.
Conditions	Have special properties that most inference blocks don't have, such as raising alarms, triggering action blocks, and producing explanations.

Action Blocks

Part III documents the palettes in the Action Blocks submenu of the Palettes menu. These palettes contain blocks that perform a variety of actions on objects in the G2 environment, including other GDA blocks. The part "Action Blocks" contains these chapters:

This chapter…	Documents blocks that…
System Actions	Operate on queues, subworkspaces, G2 icons, and sounds.
Block Actions	Operate on other GDA blocks.
Control Actions	Control how control signals flow through your diagram.
Query Actions	Let the operator choose the inference values for 1 to 4 inference paths.

Other

Part IV documents the palettes in the Other submenu of the Palettes menu. These palettes describe a variety of objects you can use with GDA. The part "Other" contains these chapters:

This chapter…	Documents objects that…
Encapsulators	Let you encapsulate part of a GDA diagram on a subworkspace.
Alarm Displays	Let you view the status of an Alarm Capability, and manipulate several Alarm Capabilities from a central location.
Path Displays	Display a data or inference path's value.
Capabilities	Add features to other GDA Blocks, such as graphing and raising alarms.
Connections	Let paths cross workspaces, join paths together, and create loops.
Network Interfaces	Let you send data to and receive data from G2 and Telewindows applications running on other machines
Rule Terminals	Let you trigger G2 rules and conclude values into rules, using GDA blocks.
Stub Tools	Let you add stubs of different types to custom blocks.

Block Descriptions

Each block in GDA has a section that contains these components:

This section…	Contains this information…
Name	The name of the block that appears on the palette.
Icon	A full-size icon of the block. All stubs and attribute displays are labeled with their names.
Description	A description of what the block does and how to configure it.
Configuring	A picture of the configuration panel for the block, and a description of each attribute label in the panel.
Example	An example of how the block works. It could show how to use the block in a diagram or show what output values it passes given certain input values. If the block's description contains many examples, this section may be missing.
See Also	References to related sections in this manual. If some of a block's features are not discussed in the description, this section contains a reference to where that feature is described. If other blocks are similar to this block, this section contains references to them.

Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names

Convention Examples	Description
true, 1.234, ok, “Burlington, MA”	G2 attribute values and values specified or viewed through dialogs
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
new-argument	User-specified values in syntax descriptions
text-string	Return values of G2 procedures and methods in syntax descriptions
File Name, OK, Apply, Cancel, General, Edit Scroll Area	GUIDE and native dialog fields, button labels, tabs, and titles
File > Save Properties	GMS and native menu choices
workspace	Glossary terms
`c:\Program Files\Gensym\`	Windows pathnames
`/usr/gensym/g2/kbs`	UNIX pathnames
`spreadsh.kb`	File names
`g2 -kb top.kb`	Operating system commands
`public void main()` `gsi_start`	Java, C and all other external code

Note Syntax conventions are fully described in the *G2 Reference Manual*.

Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure *underlined*. Each value is followed by its type:

g2-clone-and-transfer-objects
(*list*: class item-list, *to-workspace*: class kb-workspace,
delta-x: integer, *delta-y*: integer)
–> *transferred-items*: g2-list

Related Documentation

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*

- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIL Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System User's Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2-OPC Client Bridge User's Guide*
- *G2 PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*

- *G2-HLA Bridge User's Guide*
- *G2 WebLink User's Guide*

G2 JavaLink

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

G2 Diagnostic Assistant

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

To obtain customer support online:

➔ Access G2 HelpLink at `www.gensym-support.com`.

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone, fax, or email:

➔ Use the following numbers and addresses:

	Americas	Europe, Middle-East, Africa (EMEA)
Phone	(781) 265-7301	+31-71-5682622
Fax	(781) 265-7255	+31-71-5682621
Email	`service@gensym.com`	`service-ema@gensym.com`

Part I

Data Blocks

1

Entry Points

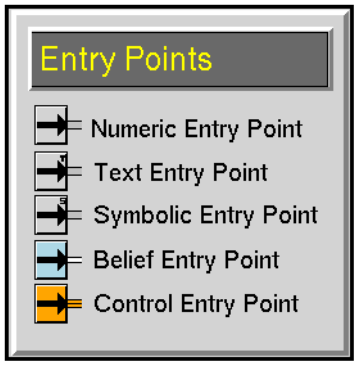
Describes the blocks you use to get data into a diagram by using sensor data from outside the application.

Introduction

The five blocks on the Entry Points palette let you enter data into your application. The Entry Points enable you to get data externally from a G2 procedure, variable, parameter, from G2 Gateway (GSI), or from an embedded variable in the table for the block.

The five blocks are: Numeric Entry Point, Text Entry Point, Symbolic Entry Point, Belief Entry Point, and Control Entry Point.

You can find these blocks on the Entry Points palette under the Data submenu of the Palettes menu:



Enabling Data Input

Entry points do not pass data until you explicitly enable data to flow.

To enable data to flow through a diagram:

➔ Select Enable Data Input from the Controls menu.

A check mark appears next to the entry, indicating that data is enabled.

To stop data from flowing through a diagram:

➔ Select Enable Data Input from the Controls menu when the menu choice is already selected.

You might want to turn data input off if you want G2 to run quicker as you modify a diagram.

Note Choosing the override menu choice from a block always passes a value, even when you disable data input.

Reading the Output Value

Entry points hold their current values in an attribute that defines a G2 variable or in the attribute sensor-value, depending on the current data source. Unlike all other blocks attributes, these attributes are displayed in the table for the block.

This table shows the attribute that defines a G2 variable for each type of entry point:

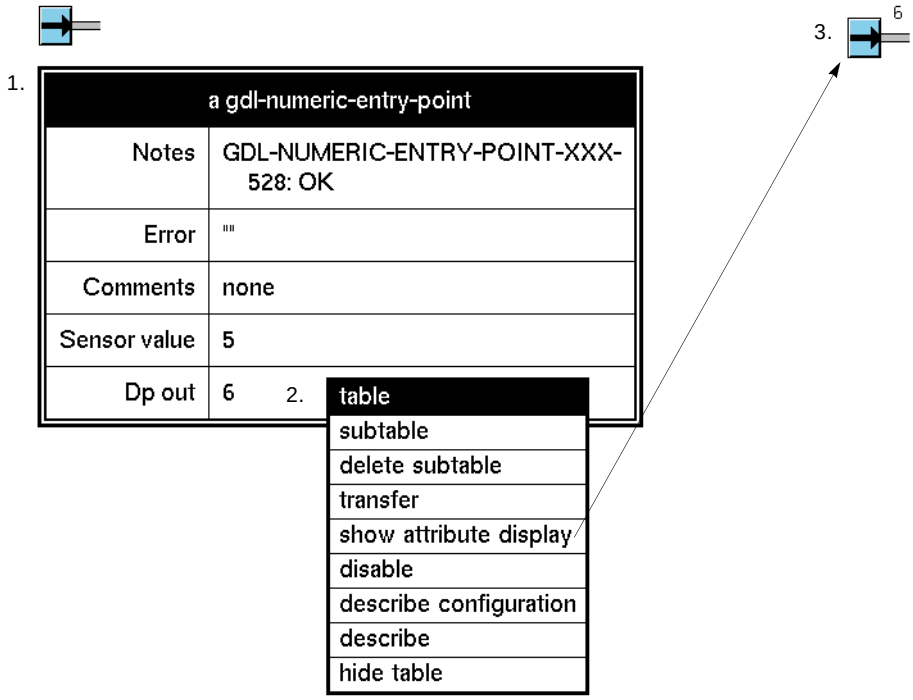
For this entry point...	The attribute of the block that stores the output value is...
Numeric Entry Point Text Entry Point Symbolic Entry Point	dp-out
Belief Entry Point	ip-out
Control Entry Point	cp-out

You can display the output value of an entry point by showing the attribute display from the table.

To see the output value beside the block:

1. Display the block's attribute table.
2. Click on the attribute's value (but not directly on the text) to display the menu.
3. Select show attribute display to display the current value.

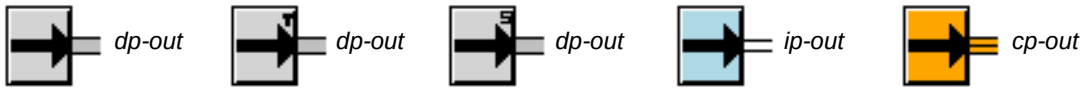
This figure shows how to show the attribute display for a Numeric Entry Point:



You change an entry point's output value by changing the value of the block's G2 variable. For information on how to change a block's output value, see "Specifying the Embedded Value for an Entry Point" on page 15.

Note If you manually override the value of an entry point, the output value in the table for the block does not show the manually overridden value.

Entry Points



The five blocks are from left to right: Numeric Entry Point, Text Entry Point, Symbolic Entry Point, Belief Entry Point, and Control Entry Point.

Any Entry Point enters data into a diagram from a G2 procedure, variable, parameter, or GSI. Any Entry Point can get its current value from one of two sources:

- From an **embedded data source**, which is a G2 variable that is an attribute of the block. You use an embedded data source when you want to obtain data from a procedure, formula, or action button, or rule. You can conclude a value directly into the G2 variable.
- From an **external data source**, which provides data directly from a G2 sensor. You use an external data source when you want to obtain data from a variable, parameter, or sensor.

The name of the attribute that contains the G2 variable that is the embedded data source depends on the type of Entry Point, as described in "Reading the Output Value" on page 4. For example, for a Numeric Entry Point, the name of this attribute is dp-out.

You configure the name of the G2 variable in the Name of Sensor attribute of the Entry Point. You configure how long the internal data is valid in the Validity Interval attribute of the entry point. You configure the data source, the formula, and the update interval of the embedded data source in the variable's subtable

You can toggle between the embedded and external data source while running your diagram to toggle between simulated and real-time data.

Using Entry Points to Obtain Data from a G2 Variable

You use Entry Points to obtain data from G2 variables. For example, you do this when you want to place all sensors for a diagram on a single workspace.

Entry Points obtain output data from one of two locations, depending on whether you are using an embedded or external data source:

- When you are using an embedded data source, the block obtains its output value from an attribute of the block, which is itself a variable. The name of this attribute depends on the type of entry point. For example, a Numeric Entry Point defines the attribute dp-out, which is an embedded variable.

- When you are using an external data source, the block obtains its output data directly from the external sensor. The Entry Point stores the current value of the external variable in an attribute of the block named sensor-value.

You can switch between these two data sources by configuring the Data Source attribute of the block. The output value of the block depends on the data source the block is using.

When you configure the attributes of the block through the configuration panel, you are configuring the *embedded* data source, namely, the attribute of the block that contains a variable, for example, dp-out. Configuring the block has no effect on the external data source.

To configure the block to use an external or embedded data source:

1. Create, name, and specify a G2 variable that supplies data to the Entry Point.

   Typically, the data server for this variable is an external data source, such as a GSI data server or G2. This variable is the external data source. You must specify the Formula and Default-update-interval attributes. You typically also specify the Validity-interval and Data-server attributes.

2. Click on the embedded variable, for example, dp-out, select the subtable menu choice, and specify the attributes the embedded variable.

   This variable is the embedded data source. You must specify the Formula and Default-update-interval attributes. You do not need to specify the Validity-interval attribute because you configure this attribute for the block. Also, you do not typically specify the Data-server attribute because the variable typically simulates real-time data in its formula.

3. Configure the Name of Sensor attribute of the Entry Point to specify the G2 variable that supplies data to the Entry Point.

   If the Data Source attribute is external, the named sensor must exist before you enter its name in the Name of Sensor. If the Data Source attribute is embedded, the named sensor does not have to exist before you enter it.

   When you configure the Name of Sensor, you are configuring the Name-of-sensor attribute in the subtable of the embedded G2 variable, for example, dp-out.

   You can specify the Name of Sensor as an expression that evaluates to a G2 variable. For more information, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.

4. Configure the Data Source to be either embedded or external.

   - Embedded means the Entry Point will use the G2 variable stored in the attribute of the block, for example, dp-out.
   - External means the Entry Point will use the external sensor whose value is stored in the Sensor-value attribute of the block.

5 Configure the Validity Interval of the Entry Point to how long the value of the embedded data source remains valid.

When you configure the Validity Interval, you are configuring the Validity-interval attribute in the subtable of the embedded G2 variable, for example, dp-out.

The default Validity Interval is supplied, which means the Entry Point uses the Validity-interval supplied by the specified data source.

Specify the Validity Interval as either a time interval, for example, 5 seconds, or indefinite, in which case the data value never expires.

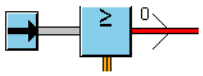
You can also override the Validity Interval given by an Entry Point by creating a rule that concludes a value directly into the dp-out output path attribute of the Entry Point, using the with expiration syntax, e.g., conclude that the dp-out of EP-1 = the current time with expiration (the current time + 5).

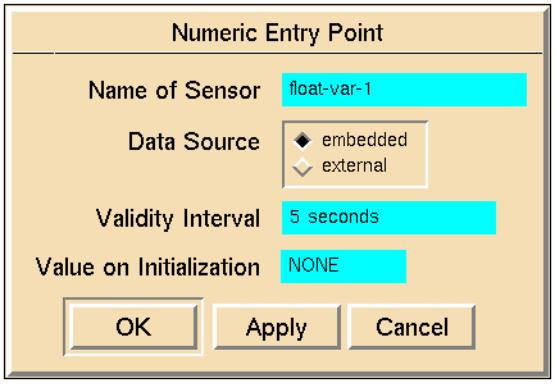
6 Specify Value on Initialization to supply a default value when the Entry Point is reset.

Note Initial values never expire, even if you specify Value on Initialization and Validity Interval for a block. Also, manual values that you provide by overriding the block never expire.

This next figure shows a Numeric Entry Point whose Name of Sensor is pointing to a variable named float-var-1, which is the external data source.

The configuration panel specifies a Validity Interval of 5 seconds, which determines the validity of the embedded data source. The default Data Source is embedded, which means the block uses the value generated by the dp-out embedded variable.





This next figure shows the external data source, the variable named float-var-1, and its associated table. The variable generates random numbers between 1 and 10 once every 20 seconds, and the data is valid for 10 seconds.

FLOAT-VAR-1

10.0, expires 20 Aug 96 2:18:12 p.m.

FLOAT-VAR-1, a float-variable	
Options	do not forward chain, breadth first backward chain
Notes	OK
Item configuration	none
Names	FLOAT-VAR-1
Tracing and breakpoints	default
Data type	float
Initial value	none
Last recorded value	10.0, expires 20 Aug 96 2:18:12 p.m.
History keeping spec	do not keep history
Validity interval	10 seconds
Formula	random (1,10)
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	inference engine
Default update interval	20 seconds

The following figure shows the table for the Numeric Entry Point, which contains the dp-out attribute and the sensor-value attribute, and the subtable for the dp-out attribute.

The dp-out attribute defines an embedded variable, which is the embedded data source for the Entry Point. The embedded variable generates random numbers between 11 and 20 once every 10 seconds, as the subtable shows.

Notice that the Validity-interval of the embedded variable corresponds to the Validity Interval attribute in the configuration panel for the block. The Sensor-value attribute shows the current value of the external data source, the variable named float-var-1.

a gdl-numeric-entry-point	
Notes	GDL-NUMERIC-ENTRY-POINT-XXX-350: OK
Error	""
Comments	none
Sensor value	4.0
Dp out	16

a gdl-quantitative-entry-point-variable, the dp out of some gdl-numeric-entry-point	
Options	do not forward chain, breadth first backward chain
Notes	OK
Item configuration	none
Names	none
Tracing and breakpoints	default
Data type	quantity
Initial value	none
Last recorded value	16, expires 20 Aug 96 2:24:00 p.m.
History keeping spec	do not keep history
Validity interval	5 seconds
Formula	random(11,20)
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	inference engine
Default update interval	10 seconds
Name of sensor	float-var-1
Quality	ok



Choosing Between Embedded and External Data Sources

By default, all Entry Points obtain their output values from the embedded data source, for example, the variable dp-out. You can cause the Entry Point to obtain its data from the external data source by reconfiguring the entry point. In this way, you can switch between simulated data that the embedded data source generates, and live data that the external data source generates.

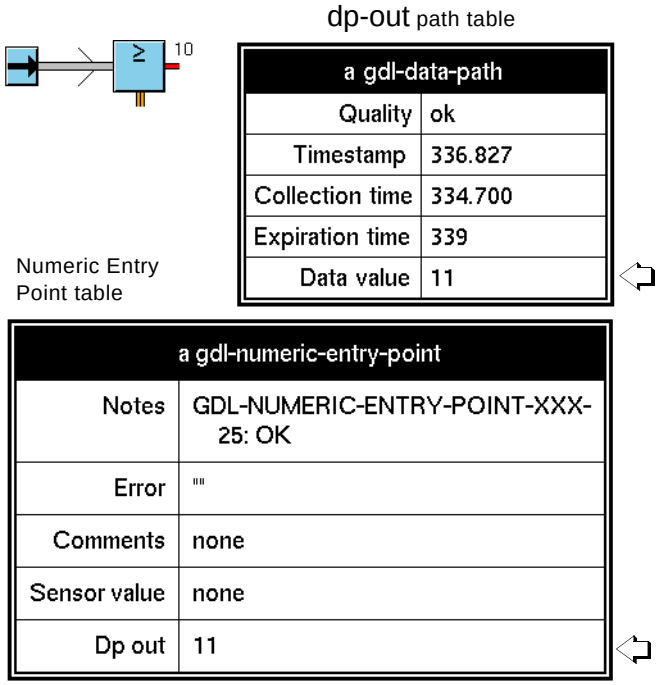
To choose between embedded and external data sources:

➔ Configure the Data Source attribute to be external or embedded.

When the block is obtaining its data from the embedded data source, for example, the dp-out variable, the arrow in the Entry Point's icon is black. When you configure the Data Source attribute to be external, the arrow in the Entry Point's icon changes to the active color of the block, whose default is cyan. In this way, you can determine the current data source.

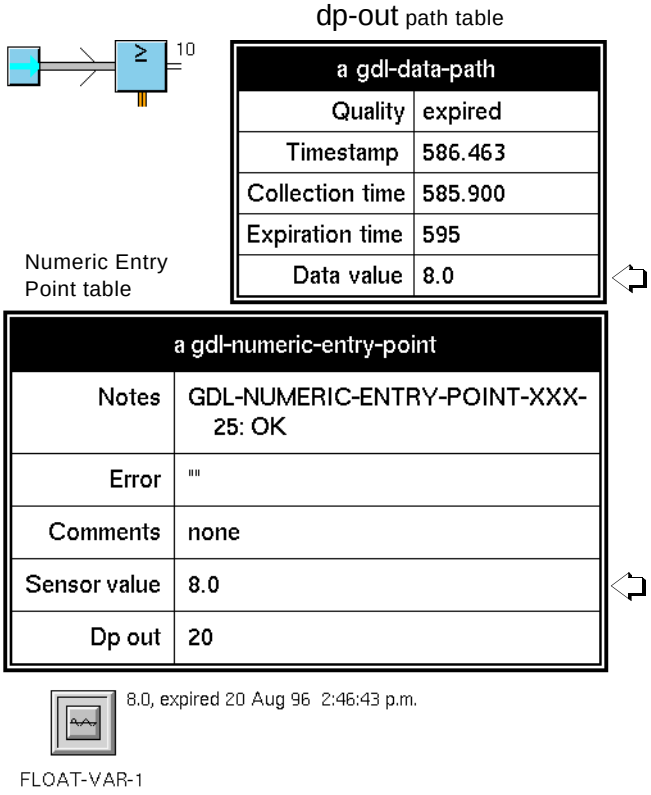
Note Even when you are using the external data source, the value of the embedded variable that is the embedded data source, for example, the dp-out, continues to update. To verify that the block is using the correct data source, display the table for the output path of the block.

The following figure shows the table for the output data path for a Numeric Entry Point when Data Source is embedded. Notice that the Data-value on the output data path of the Numeric Entry Point corresponds to the value of the dp-out embedded variable in the block's table. The output value is greater than 10, thus the inference output path of the High Value observation is true.



The following figure shows the result of configuring the Data Source attribute to be external.

Notice that the arrow on the icon of the Numeric Entry Point is now cyan, indicating it is using the external data source. The Data-value on the output data path of the Numeric Entry Point corresponds to the value of the Sensor-value attribute in the block's table, which is the current value of the float-var-1 external variable. The output value is less than 10, thus the inference output path of the High Value observation is false.



Using a G2 Variable Directly

You do not need to use an Entry Point to obtain data from a G2 variable in a diagram. Instead, you can connect variables directly to a path simply by dragging a path into the variable and making the connection. For more information on how to do this, see "Using Variables and Parameters" on page 105 in the *GDA User's Guide*.

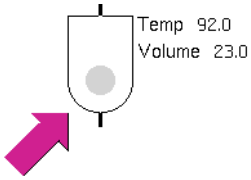
Viewing the Variable

You can view the variable that is the sensor for a particular Entry Point, using a menu choice. This menu choice is especially useful if the Entry Point and the variable are on different workspaces.

To view the variable for an Entry Point:

➔ Choose go to sensor from the Entry Point's menu.

GDA shows the workspace of the variable and places an arrow near the variable for a number of seconds. If the variable is embedded in another G2 object, the arrow points to that object, as the following figure shows.



Specifying the Embedded Value for an Entry Point

In addition to using a variable to set the output value for an Entry Point as described in "Using Entry Points to Obtain Data from a G2 Variable" on page 7, you can set the output value of the embedded variable by using:

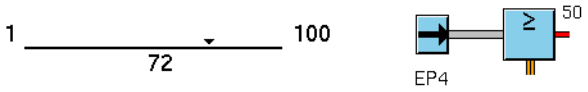
- **A button.** Any G2 button, such as sliders and action buttons, can set the value for an Entry Point.
- **A rule or procedure.** A G2 rule or procedure can conclude a value for an Entry Point.
- **A formula.** In the subtable for the Entry Point's variable, you can specify a formula that G2 evaluates when it needs the Entry Point's value.
- **Your own variable or parameter.** You can replace the Entry Point's variable with a variable or parameter created from your own object definition.

Note The Data Output block performs the opposite action of the Entry Points; it passes information *from* a diagram *to* a G2 variable or parameter.

Using a Button

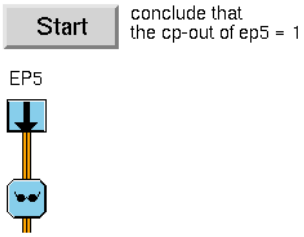
Buttons are especially useful when you are testing an application or creating a demo. They let the operator choose the value for an Entry Point. If you are using a slider, radio button, or check box, set the attribute Variable-or-parameter to the Entry Point's variable. If you are using an action button, use a conclude statement to set the Entry Point's variable.

For example, the following figure shows a slider that sets the output value for a Numeric Entry Point:



a slider	
Notes	SLIDER-XXX-1925: OK
Item configuration	none
Names	none
Label	none
Value on activation	none
Variable or parameter	the dp-out of ep4
Minimum value	1
Maximum value	100
Set value while sliding?	no
When to show value	always

In this figure, an action button starts a sequence of action blocks:



Using a Rule or Procedure

A G2 rule or procedure can set the value for an Entry Point. Whenever the rule or procedure executes, it sets the value of the Entry Point, which then passes it. To set the value of the Entry Point, use a conclude statement.

This figure shows a Belief Entry Point that gets a value from a rule. Whenever any tank has a temperature over 100, the Entry Point passes along the status value.

EP1

```
whenever the temperature T of any tank
    receives a value
and when T > 100
then conclude that the ip-out of ep1 = 1.0
```

The next figure shows a Control Entry Point that gets a value from the procedure process-bottles. When you call the procedure, the Entry Point passes along one control signal for each bottle.

EP2

```
process-bottles(num-bottles: integer)
   begin
   conclude that the cp-out of ep2 = num-
     bottles
   end
```

This figure shows a Numeric Entry Point that gets a value from the procedure adjust-speed. The procedure decrements the value of the Entry Point by an amount you specify.

EP3

```
adjust-speed(delta:float)
speed:float;
begin
   speed = gdl-get-data-path-value(ep3, the
     dp-out of ep3);
   conclude that the dp-out of ep3 = speed -
     delta;
end
```

Using a Formula

You can give a formula to an Entry Point's variable. G2 evaluates the formula at the interval specified in the variable's Default-update-interval attribute.

To specify the formula:

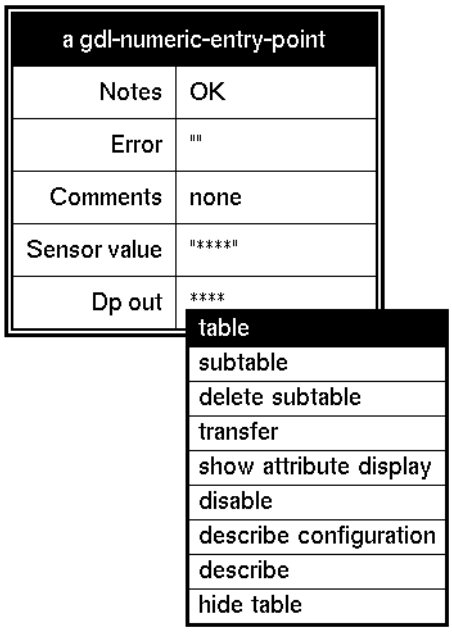
1. In the attribute table for the Entry Point, click on the attribute dp-out, ip-out, or cp-out, and select subtable from the menu.

   GDA displays the attribute's subtable.

2. Edit the attribute Formula in the subtable.

3 Set the attribute Default-update-interval to the interval at which you want GDA to evaluate the formula.

1. 



2.and 3.

a gdl-quantitative-entry-point-variable, the dp out of some gdl-numeric-entry-point	
Options	do not forward chain, breadth first backward chain
Notes	the dp out of GDL-NUMERIC-ENTRY-POINT-XXX-451: OK
Item configuration	none
Names	none
Tracing and breakpoints	default
Data type	quantity
Initial value	none
Last recorded value	no value
History keeping spec	do not keep history
Validity interval	indefinite
Formula	average(the volume of t1, the volume of t2, the volume of t3)
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	inference engine
Default update interval	10 seconds
Name of sensor	none
Quality	ok

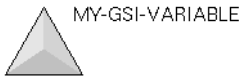
Using Your Own Variable Definition

You can replace the variable in an Entry Point with a variable created from your own object definition. This method is especially useful when the you would like to add attribute to the Entry Point or change the inheritance of the embedded variable.

To use your own variable definition:

1 Create the variable definition.

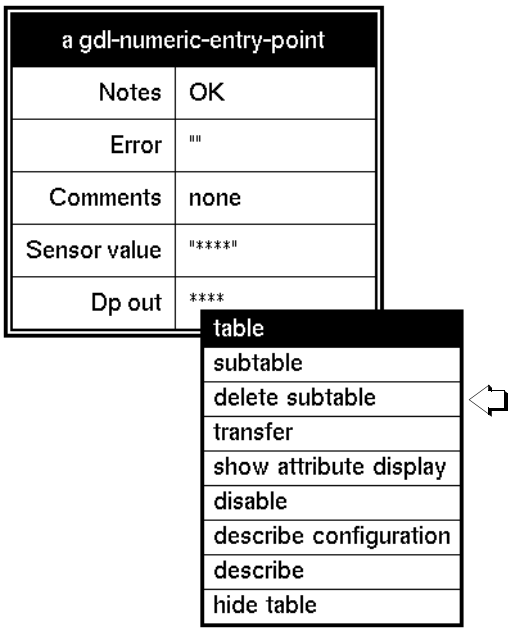
This figure shows a completed attribute table for a variable definition that uses GSI:



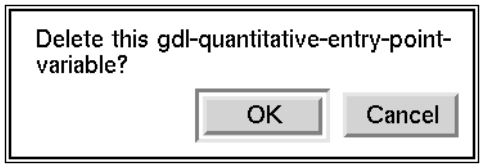
MY-GSI-VARIABLE, an object-definition	
Notes	OK
User restrictions	none
Class name	my-gsi-variable
Superior class	float-variable
Attributes specific to class	none
Capabilities and restrictions	gsi-data-service
Class restrictions	none
Change	none
Menu option	a final menu choice
Inherited attributes	none
Default settings	none
Attribute displays	inherited
Stubs	inherited
Color	inherited
Icon description	inherited

2 Go into Administrator mode.

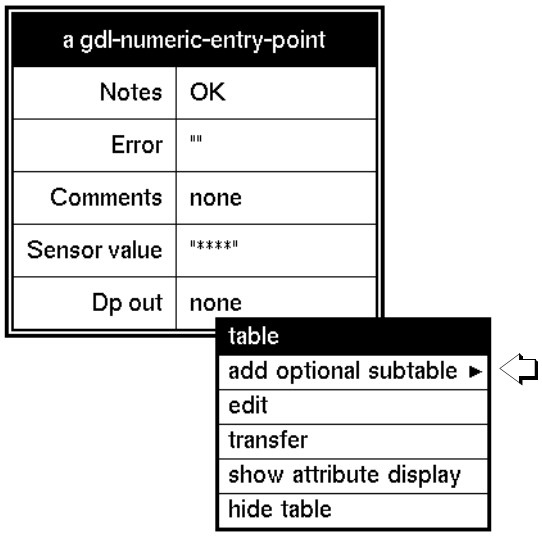
3 Delete the subtable for the Entry Point's variable by clicking on the variable in the Entry Point's attribute table, and selecting delete subtable from the menu that appears:



4 Click OK in the dialog that G2 displays, asking you to confirm that you want to delete the subtable.



5 Add a new subtable for your variable definition by clicking on the variable in the Entry Point's attribute table, and selecting add optional subtable from the menu that appears:



G2 displays a menu asking you to choose a class.

6 Choose g2-variable from the menu, and follow the menu hierarchy down until you see the name of the superior class used to define the new variable class specified in Step 1.

The final menu contains your new variable definition. Click on the name of your variable definition.

G2 displays the subtable for your variable.

7 Edit the variable to suit your needs.

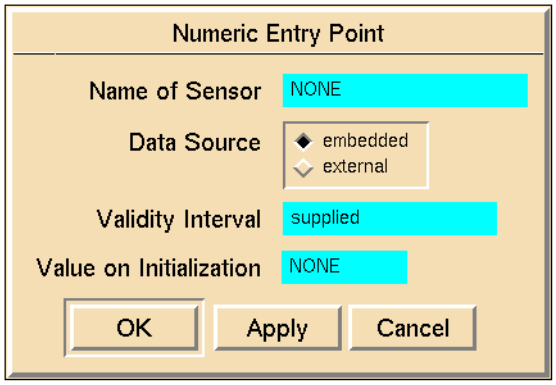
a my-gsi-variable, the dp out of some gdl-numeric-entry-point	
Options	do not forward chain, breadth first backward chain
Notes	OK
User restrictions	none
Names	none
Tracing and breakpoints	default
Data type	float
Initial value	none
Last recorded value	no value
History keeping spec	do not keep history
Validity interval	supplied
Formula	none
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	GSI data server
Default update interval	none
Gsi interface name	none
Gsi variable status	0

Using Symbolic Entry Points

You can connect a Symbolic Entry Point to only one type of block: the Equality Observation. For an example of how it is used, see "Example" on page 27.

Configuring

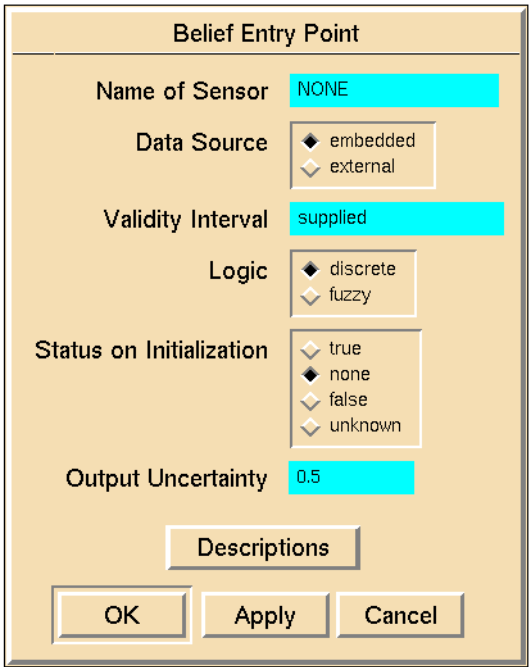
This is the configuration panel for the Numeric Entry Point. The panel for the Text Entry Point and Symbolic Entry Point is identical except for the block name.



Attribute	Description
Name of Sensor	The name of the G2 variable that supplies data to the Entry Point. The sensor you specify must exist when Data Source is external. For information on how to use an expression for the Name of Sensor, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.
Data Source	Determines whether the entry point obtains its output value from the embedded variable, which is an attribute of the entry point, for example, dp-out, or from an external variable, which is the value of the Name of Sensor attribute. When Data Source is external, the arrow on the icon changes to the active color for blocks.

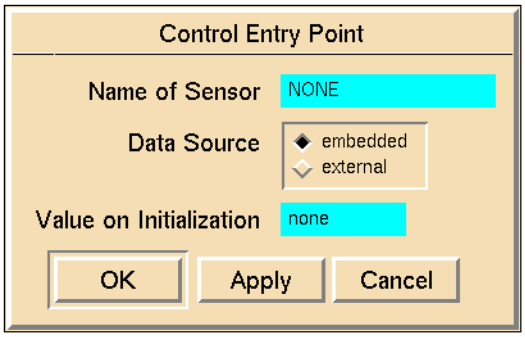
Attribute	Description
Validity Interval	The amount of time that the current value of the Entry Point remains valid, specified either as a time interval, for example, 5 seconds, or indefinite, in which case the data value never expires. The default Validity Interval is supplied, which means the Entry Point uses the Validity-interval supplied by the Name of Sensor variable. The specification of this attribute overrides the Validity-interval given by the variable.
Value on Initialization	See "Specifying Initial Values" on page 83 in the *GDA User's Guide*.

This is the configuration panel for the Belief Entry Point.



Attribute	Description
Name of Sensor	The name of the G2 variable that supplies data to the Entry Point.
Data Source	Determines whether the entry point obtains its output value from the embedded variable, which is an attribute of the entry point, for example, dp-out, or from an external variable, which is the value of the Name of Sensor attribute. When Data Source is external, the arrow on the icon changes to the active color for blocks.
Validity Interval	The amount of time that the current value of the Entry Point remains valid, specified either as a time interval, for example, 5 seconds, or indefinite, in which case the data value never expires. The default Validity Interval is supplied, which means the Entry Point uses the Validity-interval supplied by the Name of Sensor variable. The specification of this attribute overrides the Validity-interval given by the variable.
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown	See "Specifying and Generating Explanations" on page 93 in the *GDA User's Guide*.

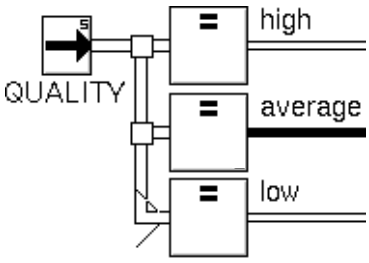
This is the configuration panel for the Control Entry Point.



Attribute	Description
Name of Sensor	The name of the G2 variable that supplies data to the Entry Point.
Data Source	Determines whether the entry point obtains its output value from the embedded variable, which is an attribute of the entry point, for example, dp-out, or from an external variable, which is the value of the Name of Sensor attribute. When Data Source is external, the arrow on the icon changes to the active color for blocks.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

This diagram uses three equality blocks to choose among three paths, depending on the value of a Symbolic Entry Point:



See Also

For general information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 4	*Reference Manual*

2

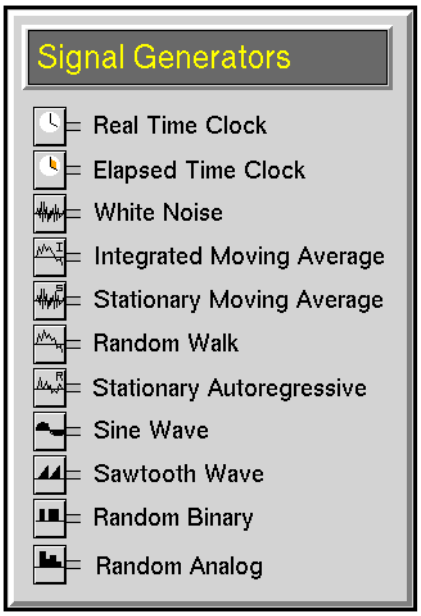
Signal Generators

Describes the blocks you use to get data into a GDA diagram by simulating a signal.

Introduction

The Signal Generators palette contains blocks that simulate real-time data, for example, Sine Wave, Sawtooth Wave, etc.

You find the Signal Generators palette under the Data submenu of the Palettes menu:



Enabling Data Input

Signal generators do not pass data until you explicitly enable data to flow. After you enable data input, these blocks immediately update their output values, then update their value regularly.

To enable data to flow through a GDA diagram:

➔ Select Enable Data Input from the Controls menu.

A check mark appears next to the menu choice when it is selected.

To stop data from flowing through a diagram:

➔ Select Enable Data Input from the Controls menu when it is checked.

The check mark will disappear indicating that data input is no longer enabled.

You might want to disable data input if you want G2 to run more quickly as you modify a diagram.

Note Choosing the override menu choice from a block always passes a value, even when you disable data input.

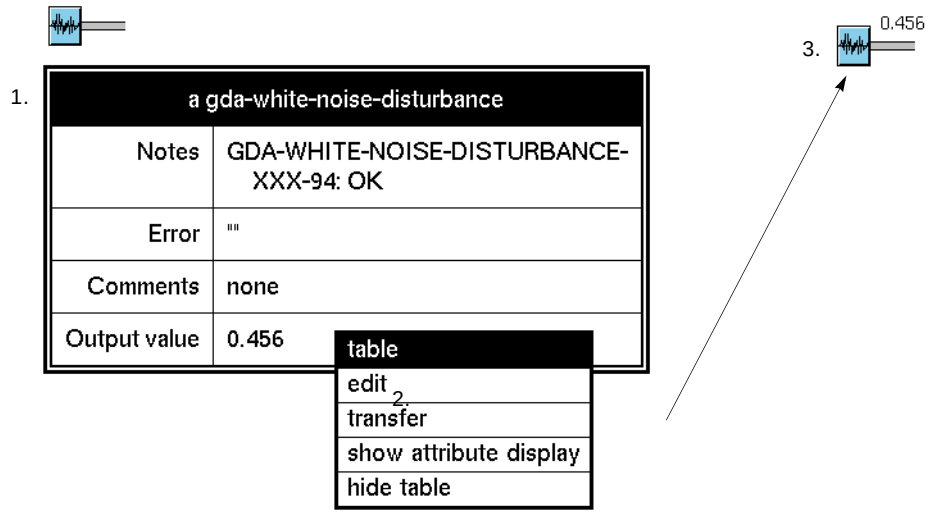
Reading the Output Value

Signal generators hold their current values in an attribute of the block named output-value.

To see the output value beside the block:

1. Display the block's attribute table.
2. Click on the name of the output-value attribute to display the menu.
3. Select show attribute display to display the current value.

This figure shows how to show the attribute display for a White Noise block:



Specifying How Often to Generate Values

The attribute Sample Period specifies how often a signal generator passes a new value. If this value is large, the block produces a coarser signal and your application runs faster. By default, the value is 5 seconds.

For example, if a Real Time Clock block has a Sample Period of 1 second, it might send out these values over 10 seconds: 4001, 4002, 4003, 4004, 4005, 4006, 4007, 4008, 4009, and 4010. If the block has a Sample Period of 2 seconds, it would send out the following values over the same period of time: 4002, 4004, 4006, 4008, and 4010.

If you change a block's Sample Period while your application is running, GDA resets the block and starts passing values according to the new period.

Generating a Signal

The following blocks generate data to let you simulate values and test your application. There are three general kinds:

- Time signal blocks generate values based on the time.
- Random signal blocks generate random values.
- Periodic signal blocks pass values based on a function that repeats cyclically.

Time Signal Blocks

The following blocks pass times. They are especially useful for timing events or producing a sequence of steadily increasing numbers.

- The Real Time Clock block on page 34 passes the number of seconds that G2 has been running, in real time.
- The Elapsed Time Clock block on page 36 passes the number of seconds since it was created or last reset.

Random Signal Blocks

The following blocks generate random numbers. Their current values do not depend on their previous values.

- The White Noise block on page 38 generates values that are normally distributed, with a mean and variance you specify. This block is especially useful when you add it to another signal to simulate noise.
- The Random Analog block page 60 generates values that are evenly distributed between a minimum and maximum which you specify.
- The Random Binary block page 60 generates one of two values that you specify. Usually, the values are 0 or 1. This block is especially useful for simulating whether a piece of equipment is either on or off.

Statistical Signal Blocks

These blocks generate disturbances commonly found in process applications. The Integrated Moving Average block and the Stationary Moving Average block produce signals that most closely simulate real-life signals.

- The Integrated Moving Average block on page 41 simulates the output from a sensor that is measuring a Random Walk signal in which the sensor adds its own noise to the signal. For example, the quality of a piece of equipment that is slowly aging may follow this pattern.
- The Stationary Moving Average block on page 44 is useful for simulating such things as the quality of raw materials. The quality varies within certain known

limits, and the quality of one batch helps predict the quality of the next batch. For example, batches of lumber from the same forest can have similar quality.

- The Random Walk block on page 47 produces a series of values in which the difference between successive values varies around a mean, and the difference between one pair of values does not depend on the difference between the previous pair.
- The Stationary Autoregressive Disturbance block on page 50 produces a random sequence of numbers which are normally distributed around a mean value, in which the current value depends on the previous value.

Periodic Signal Blocks

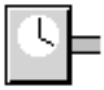
These blocks generate periodic signals. They produce values from functions that repeat cyclically. Their attributes let you change the function's shape.

- The Sine Wave block on page 53 generates sine values.
- The Sawtooth Wave block on page 56 generates values that ramp from a minimum to a maximum value and then immediately drop back to the minimum value.

Setting the Period of Signal Generators

You should always set the Period attribute for a Sine Wave or Sawtooth Wave to a number greater than 0.

Real Time Clock


dp-out

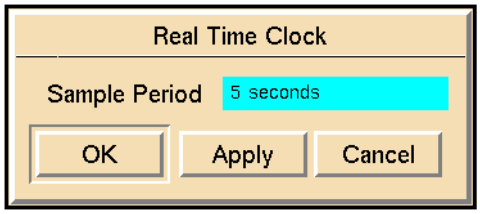
The Real Time Clock passes the number of seconds that G2 has been running. This is the same value that the G2 expression the current real time returns.

Note This block passes the real time. If you are running G2 with Scheduler-mode set to as fast as possible, the real time is not the same as the G2 time. Use the Data Time Stamp block on page 152 to get the G2 time.

You can use this block to time events or to produce a sequence of steadily increasing numbers.

Configuring

This is the configuration panel for the Real Time Clock.



Attribute	Description
Sample Period	See "Specifying How Often to Generate Values" on page 31.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*

For more information on...	See...	In this book...
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*
The Data Time Stamp block	page 152	*Reference Manual*

Elapsed Time Clock

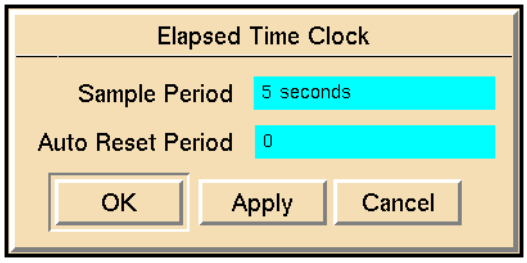
dp-out

The Elapsed Time Clock passes the number of seconds since it was created or last reset. To reset this block to zero at a regular interval, use the attribute Auto Reset Period. The block resets itself whenever the Auto Reset Period passes. If Auto Reset Period is 0, the block never resets itself.

You can use this block to time events or to produce a sequence of steadily increasing numbers.

Configuring

This is the configuration panel for the Elapsed Time Clock.



Attribute	Description
Sample Period	See "Specifying How Often to Generate Values" on page 31.
Auto Reset Period	How often the block resets. An Auto Reset Period of 0 means the block never resets. The Auto Reset Period should be greater than the Sample Period.

Example

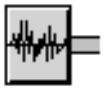
If the block's Sample Period is 1 second and Auto Reset Period is 5 seconds, it passes the values 0.0, 1.0, 2.0, 3.0, 4.0, 0.0, 1.0, and so on.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*
The Data Time Stamp block	page 152	*Reference Manual*

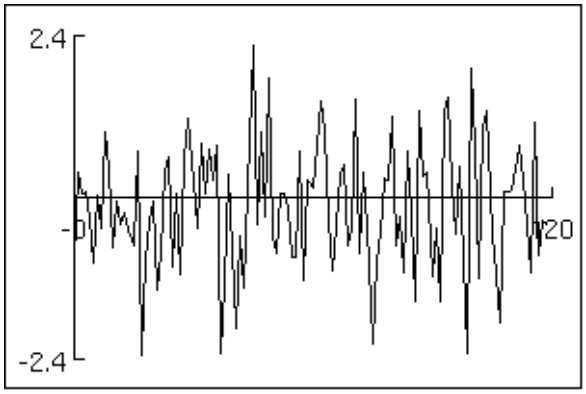
White Noise


dp-out

The White Noise block generates random values that are normally distributed around a mean. Its current output value does not depend on any previous output value. This block is especially useful when you add it to another signal to simulate noise.

Note Like the White Noise block, the Random Analog block on page 60 also produces random values. However, the Random Analog block produces values that are *evenly* distributed between a minimum and maximum you specify. The White Noise block produces values that are *normally* distributed with a mean and variance you specify. (That is, the values are concentrated in the middle of the range.)

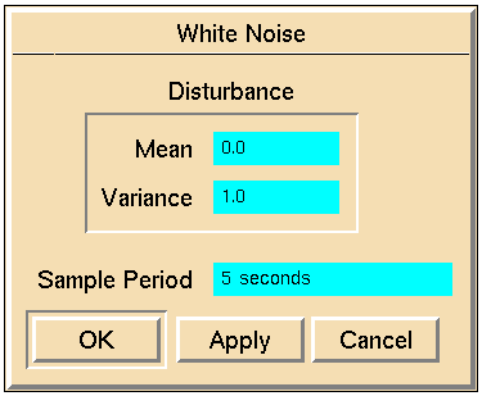
This figure shows the output from a White Noise block. The Disturbance Mean is 0 and the Disturbance Variance is 1.0.



To specify the range of output values, set the attributes Disturbance Mean and Disturbance Variance. Disturbance Mean is the mean output value, and Disturbance Variance is the variance of the output values.

Configuring

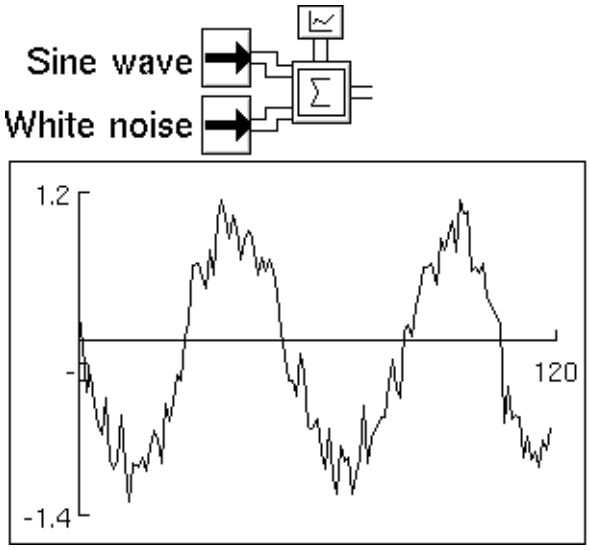
This is the configuration panel for the White Noise block.



Attribute	Description
Disturbance Mean	The mean output value of the block.
Disturbance Variance	The variance of the output values of the block.
Sample Period	How often the block passes a new value.

Example

The White Noise block in this figure adds noise to a Sine Wave signal. The Disturbance Mean is 0.0 and the Disturbance Variance is 0.025.



See Also

For more information on how to use this block, see the sections below.

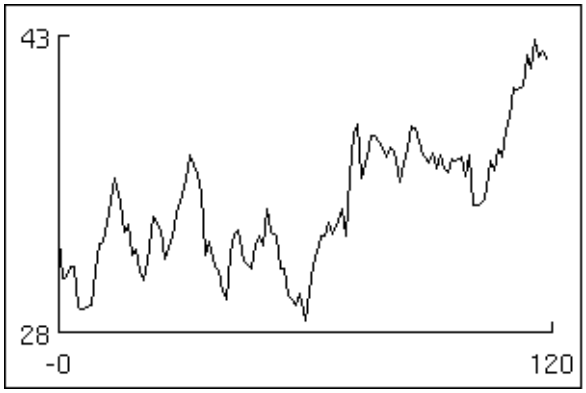
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

Integrated Moving Average

The Integrated Moving Average block produces values in which the difference between successive values varies around a mean, and the difference between one pair of values depends on the difference between the previous pair. This block produces values that do not change much from one value to the next, but that can vary greatly over time.

This signal is common in process control applications. It simulates the output from a sensor that is measuring a Random Walk signal, and the sensor adds its own noise to the signal.

This figure shows the output from an Integrated Moving Average block. The Disturbance Mean is 0.0, the Disturbance Variance is 1.0, and the Disturbance Filter Constant is 0.8.



Setting the Signal's Attributes

To specify how big the difference can be between two successive pairs of values, set the attributes Disturbance Mean and Disturbance Variance. Disturbance Mean is the mean difference, and Disturbance Variance is the variance of the differences.

To specify how much the difference between two values affects the difference between the next two values, set the attribute Disturbance Filter Constant to a number from 0.0 to 1.0. A common value is 0.8.

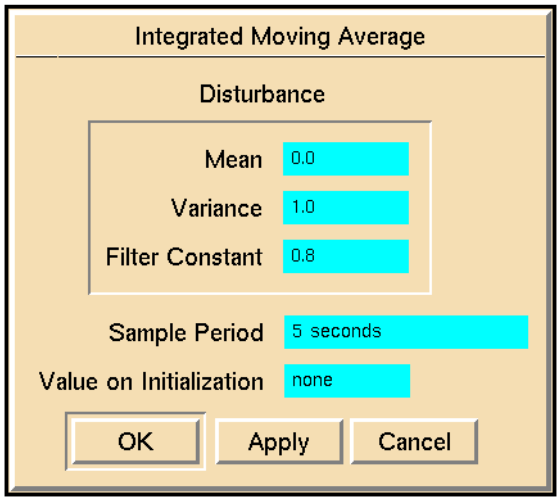
The Integrated Moving Average block uses this equation to compute its output value. The variables in the equation are explained in the following table.

$$output_n = output_{n-1} + rand - \phi(output_{n-2} - output_{n-1})$$

This variable…	Is…
$output_n$	The current output value
$output_{n-1}$	The previous output value
$output_{n-2}$	The output value before the previous output value
rand	One value in a series that has a mean of Disturbance Mean and a variance of Disturbance Variance
ϕ	The attribute Disturbance Filter Constant

Configuring

This is the configuration panel for the Integrated Moving Average block.



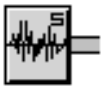
Attribute	Description
Disturbance Mean	The mean difference between two successive pairs of values.
Disturbance Variance	The variance of the difference between two successive pairs of values.
Disturbance Filter Constant	A number between 0.0 and 1.0, which determines the amount by which the difference between two values affects the difference between the next two values.
Sample Period	How often the block passes a new value.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

See Also

For more information on how to use this block, see the sections below

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

Stationary Moving Average

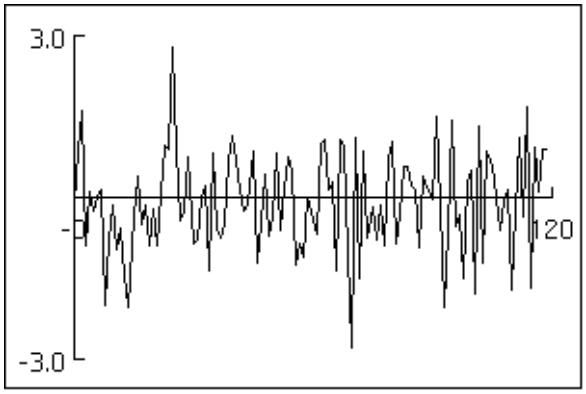


dp-out

The Stationary Moving Average block produces values that vary around a mean, in which the difference between one pair of values depends on the difference between the previous pair of values. It is like a White Noise signal, except that the difference between one pair of values affects the difference between the next pair.

This signal is useful for simulating such things the quality of raw materials. The quality varies within certain known limits, and the quality of one batch helps predict the quality of the next batch. (For example, batches of lumber from the same forest may have similar quality.)

This figure shows the output from a Stationary Moving Average block. The Process Mean is 0.0, the Disturbance Mean is 0.0, the Disturbance Variance is 1.0, and the Disturbance Filter Constant is 0.8.



Setting the Signal's Attributes

To specify what the average value should be, set the attribute Process Mean.

To specify how big the difference can be between two successive pairs of values, set the attributes Disturbance Mean and Disturbance Variance. Disturbance Mean is the mean difference, and Disturbance Variance is the variance of the differences.

To specify how much the difference between two values affects the difference between the two next values, set the attribute Disturbance Filter Constant to a number from 0.0 to 1.0. A common value is 0.8.

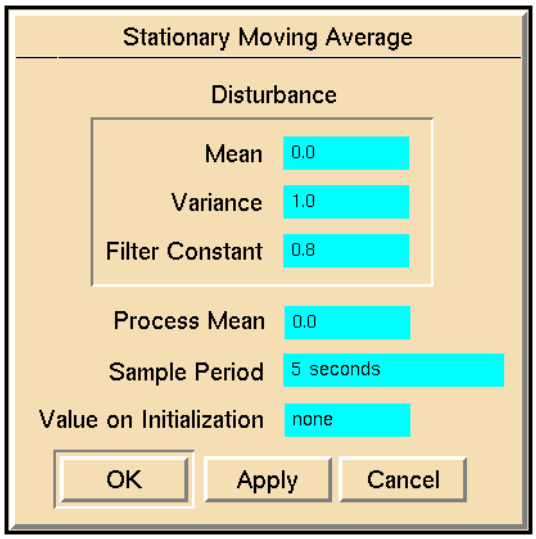
The Stationary Moving Average block uses this equation to compute its output value. The variables in the equation are explained in the table that follows. *Process-mean* is the value of the Process Mean attribute of the block.

$$\text{output}_n = \textit{Process-mean} + \text{rand} - \phi(\text{output}_{n-2} - \text{output}_{n-1})$$

This variable…	Is…
output_n	The current output value
output_{n-1}	The previous output value
output_{n-2}	The output value before the previous output value
rand	One value in a series that has a mean of Disturbance Mean and a variance of Disturbance Variance
ϕ	The attribute Disturbance Filter Constant

Configuring

This is the configuration panel for the Stationary Moving Average block.



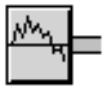
Attribute	Description
Disturbance Mean	The mean difference between two successive pairs of values.
Disturbance Variance	The variance of the difference between two successive pairs of values.
Disturbance Filter Constant	A number between 0.0 and 1.0, which determines the amount by which the difference between two values affects the difference between the next two values.
Process Mean	The mean output value of the block.
Sample Period	How often the block passes a new value.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide.*

See Also

For more information on how to use this block, see the sections below

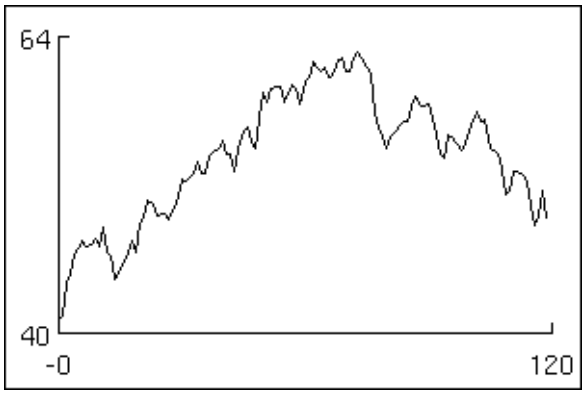
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

Random Walk



The Random Walk block produces a series of values in which the difference between successive values varies around a mean, and the difference between one pair of values does not depend on the difference between the previous pair. The values themselves vary widely, but the difference between one value and the next is restricted.

This figure shows the output from a Random Walk block. The Disturbance Mean is 0.0, and the Disturbance Variance is 1.0.



Setting the Signal's Attributes

To specify the range of output values, set the attributes Disturbance Mean and Disturbance Variance. Disturbance Mean is the mean output value, and Disturbance Variance is the variance of the output values.

The Random Walk block uses this equation to compute its output value. The variables in the equation are explained in the table that follows.

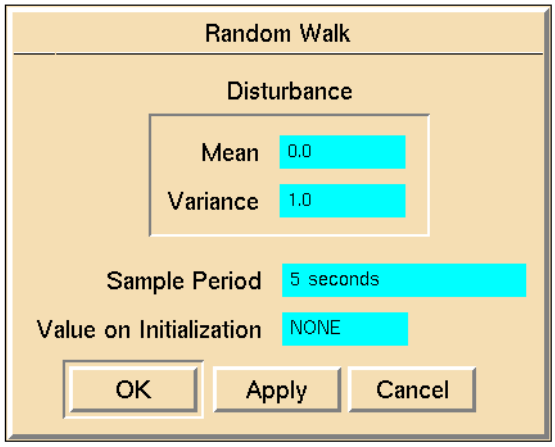
$$\text{output}_n = \text{output}_{n-1} + \text{rand}$$

This variable…	Is…
output_n	The current output value

This variable...	Is...
$output_{n-1}$	The previous output value
rand	One value in a series that has a mean of Disturbance Mean and a variance of Disturbance Variance

Configuring

This is the configuration panel for the Random Walk block.



Attribute	Description
Disturbance Mean	The mean output value of the block.
Disturbance Variance	The variance of the output value of the block.
Sample Period	How often the block passes a new value.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide.*

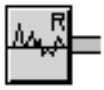
See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*

For more information on...	See...	In this book...
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

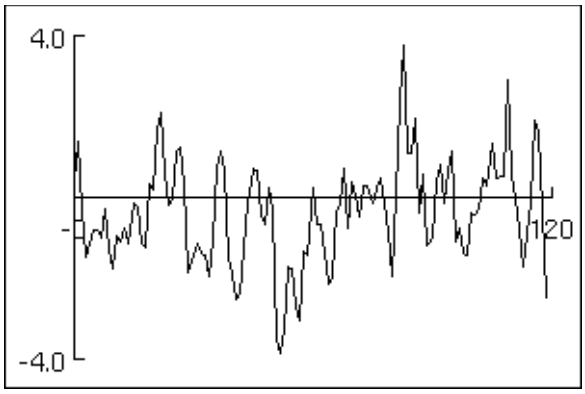
Stationary Autoregressive Disturbance



dp-out

The Stationary Autoregressive Disturbance block produces a random sequence of numbers which are normally distributed around a mean value, in which the current value depends on the previous value. This block produces output similar to what you would get from connecting a White Noise block to a First-Order Exponential Filter.

This figure shows the output from a Stationary Autoregressive Disturbance block. The Process Mean is 0.0, the Disturbance Mean is 0.0, the Disturbance Variance is 1.0, and the Disturbance Filter Constant is 0.8.



Setting the Signal's Attributes

To specify what the average value should be, set the attribute Process Mean.

To specify the range of output values, set the attributes Disturbance Mean and Disturbance Variance. Disturbance Mean is the mean output value, and Disturbance Variance is the variance of the output values.

To specify how much the current value depends on the previous value, set the attribute Disturbance Filter Constant to a number from 0.0 to 1.0.

The Stationary Autoregressive block uses this equation to compute its output value. The variables in the equation are explained in the table that follows.

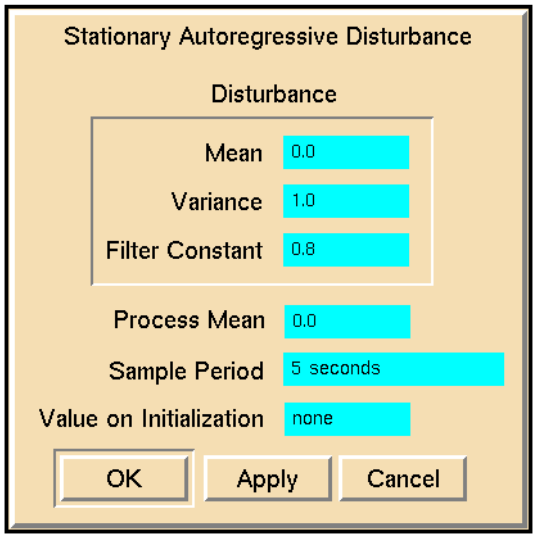
.

$$\text{output}_n = \textit{Process-mean} + \text{rand} + \phi(\text{output}_{n-1})$$

This variable…	Is…
$output_n$	The current output value
$output_{n-1}$	The previous output value
rand	One value in a series that has a mean of Disturbance Mean and a variance of Disturbance Variance
ϕ	The attribute Disturbance Filter Constant

Configuring

This is the configuration panel for the Stationary Autoregressive Disturbance block.



Attribute	Description
Disturbance Mean	The mean output value of the block.
Disturbance Variance	The variance of the output value of the block.
Disturbance Filter Constant	A number between 0.0 and 1.0, which determines the amount by which the difference between two values affects the difference between the next two values.
Process Mean	The mean output value of the block.

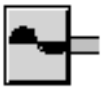
Attribute	Description
Sample Period	How often the block passes a new value.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide.*

See Also

For more information on how to use this block, see the sections below

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

Sine Wave



The Sine Wave block generates sine values. It is a periodic signal generator and repeats its pattern of outputs cyclically.

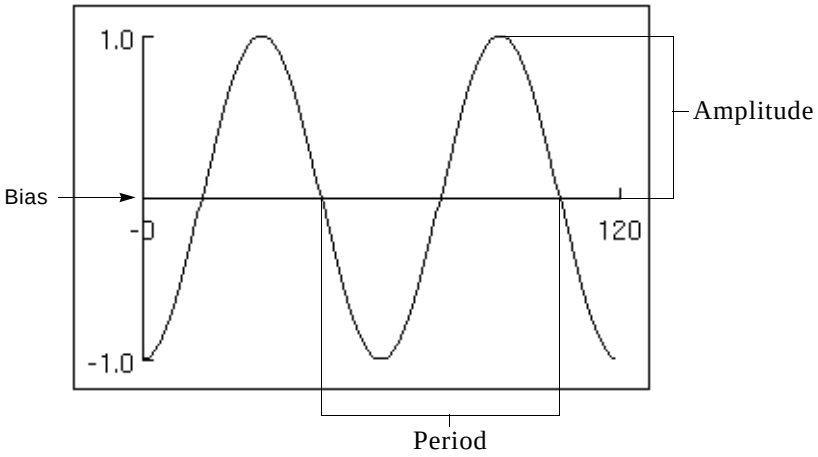
Specifying the Shape

The attributes Amplitude, Bias, and Period specify the shape of the sine wave. The Amplitude is half the difference between the minimum and maximum values. The Bias is the mean value between the minimum and the maximum. The Period is the amount of time that the block takes to complete a cycle.

Specifying a Phase

The Phase Angle determines where the Sine Wave block starts its cycle. It is a number of degrees between 0 and 360. For example, a Sine Wave block with a Phase Angle of 0 starts its cycle at the Bias going towards the maximum value. A Sine Wave block with a Phase Angle of 90 starts its cycle at the maximum value.

This figure shows a graph of a sine wave with a Period of 60, an Amplitude of 1.0, and a Bias of 0.0, and a Phase of 270.



Note Another way to change the place where the block starts its cycle is to set the attribute Reset Phase to yes and choose reset from the block's menu. For more information see "Resetting" below.

Resetting

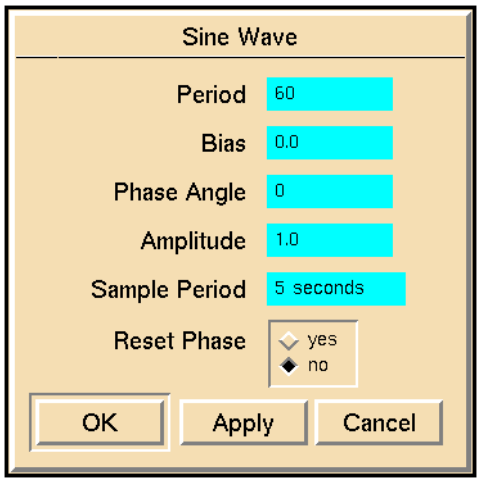
The attribute Reset Phase determines what happens when you choose reset from the block's menu:

If Reset Phase is…	The block does this when you choose reset …
yes	Returns to the beginning of the wave's cycle.
no	Continues as before.

If you change Phase Angle as GDA is running, GDA uses the new value the next time you reset the block.

Configuring

This is the configuration panel for the Sine Wave block with its default values.



Attribute	Description
Period	The amount of time that the block takes to complete a cycle.
Bias	The value between the minimum and the maximum.
Phase Angle	A number of degrees between 0 and 360, which determines where the Sine Wave block starts its cycle.

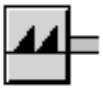
Attribute	Description
Amplitude	Half the difference between the minimum and maximum values.
Sample Period	How often the block passes a new value.
Reset Phase	Whether the block returns to the beginning of the cycle when reset (the default) or continues where the signal left off.

See Also

For more information on how to use this block, see the sections below

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

Sawtooth Wave



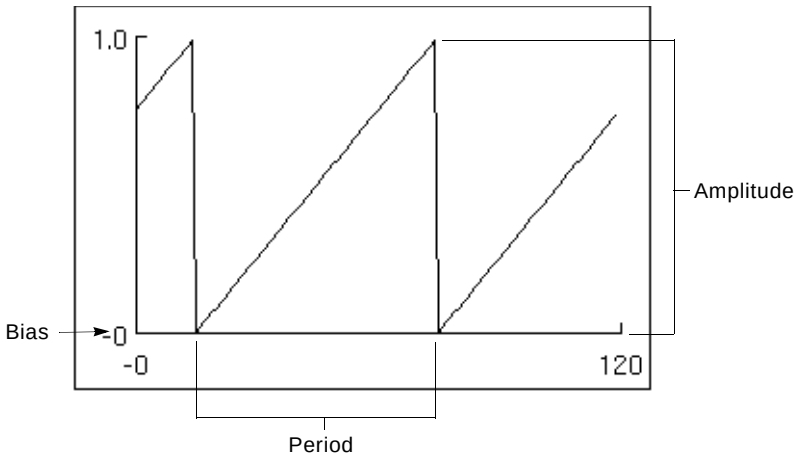
dp-out

The Sawtooth Wave block generates Sawtooth values. It gradually rises from its minimum to maximum values and then immediately drops back to its minimum value. It is a periodic signal generator and repeats its pattern of outputs cyclically.

Specifying the Shape

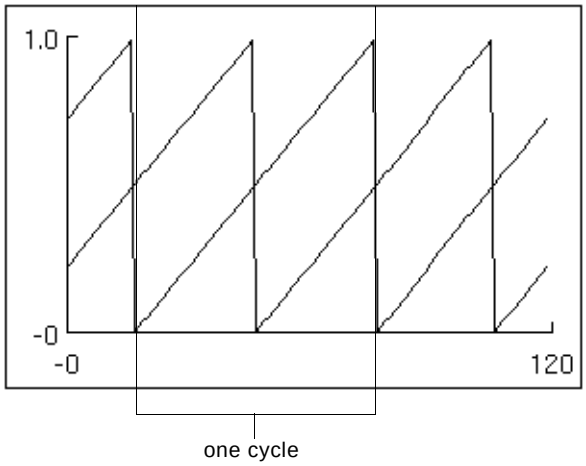
The attributes Amplitude, Bias, and Period specify the shape of the sawtooth wave. The Amplitude is the difference between the minimum and maximum values. The Bias is the minimum value. The Period is the amount of time that the block takes to complete a cycle, that is, to rise from the minimum value, gradually increase to the maximum value, and suddenly drop to the minimum.

This figure shows a graph of a sawtooth wave with a Period of 60, an Amplitude of 1.0, and a Bias of 0.0.



Specifying a Phase

The Phase Angle determines where the Sawtooth Wave block starts its cycle. It is a number of degrees between 0 and 360. For example, a Sawtooth Wave block with a Phase Angle of 0 starts its cycle at the Bias. A Sawtooth Wave block with a Phase Angle of 90 starts its cycle at a value between the minimum and maximum values. This figure shows a graph of two sawtooth waves with the same values for Period, Bias, and Amplitude but with different values for Phase Angle. One is 0 and the other is 90.



Note Another way to change the place where the block starts its cycle is to set the attribute Reset Phase to yes and choose reset from the block's menu. For more information see "Resetting" below.

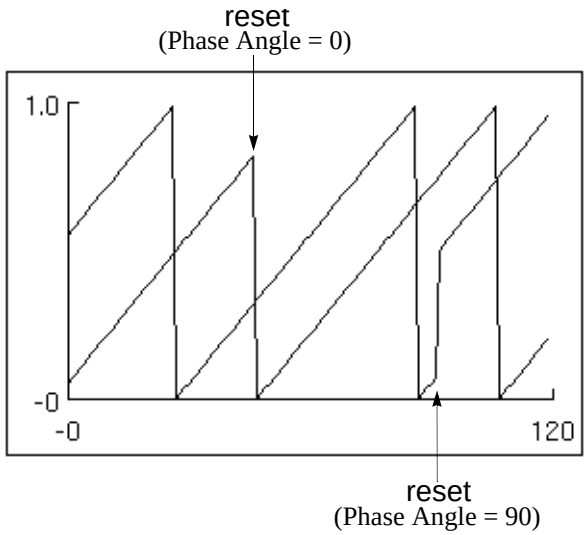
The time at which you create or initialize a Sawtooth Wave block does not change the place at which the block starts its cycle. Only the Phase Angle does. For example, if you initialize two Sawtooth Wave blocks at different times but set all their attributes alike, including the Phase Angle, they will always be at the same spot in their cycle.

Resetting

The attribute Reset Phase determines what happens when you choose reset from the block's menu. This table describes the possibilities:

If Reset Phase is…	The block does this when you choose reset …
yes	Returns to the beginning of the wave's cycle.
no	Continues as before.

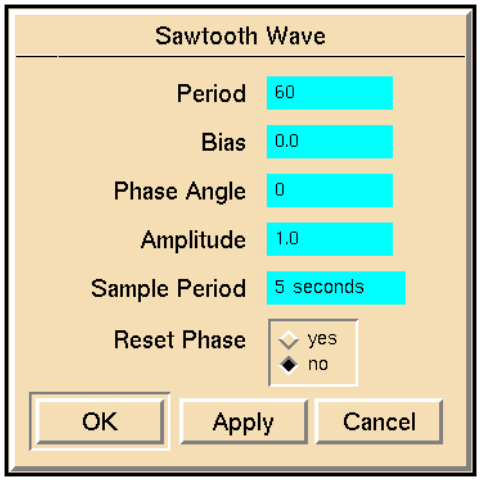
This figure shows a graph of two Sawtooth Wave blocks, with Reset Phase set to yes, that have been reset. One has a Phase Angle of 0, and the other of 90.



If you change Phase Angle as GDA is running, GDA uses the new value the next time you reset the block.

Configuring

This is the configuration panel for the Sawtooth Wave block.



Attribute	Description
Period	The amount of time that the block takes to complete a cycle.
Bias	The minimum value of the cycle.

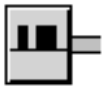
Attribute	Description
Phase Angle	A number of degrees between 0 and 360, which determines where the Sawtooth Wave block starts its cycle.
Amplitude	The difference between the minimum and maximum values.
Sample Period	How often the block passes a new value.
Reset Phase	Whether the block returns to the beginning of the cycle when reset (the default) or continues where the signal left off.

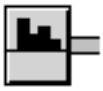
See Also

For more information on how to use this block, see the sections below

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

Random Binary, Random Analog

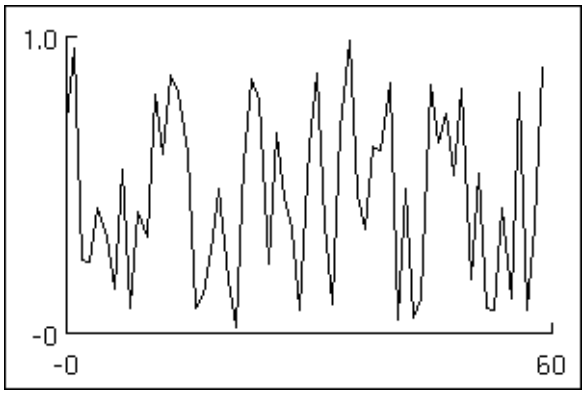


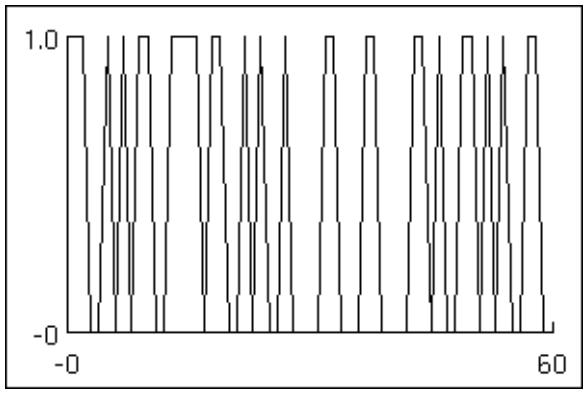


These blocks produce random values. Specify the values to produce with the attributes Min Value and Max Value. The Random Binary block produces values that are either Min Value or Max Value. The Random Analog block produces values within the range from Min Value to Max Value.

Note Like the Random Analog block, the White Noise block on page 38 also produces random values. However, the Random Analog block produces values that are *evenly* distributed between a minimum and maximum you specify. The White Noise block produces values that are *normally* distributed with a mean and variance you specify (that is, the values are concentrated in the middle of the range).

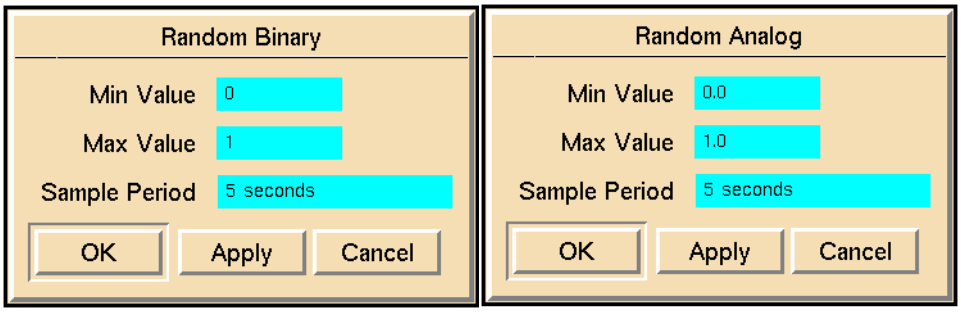
This figure shows the output from a Random Analog block, and the following figure shows the output from a Random Binary block. In both cases, the Min Value is 0.0 and the Max Value is 1.0.





Configuring

These are the configuration panels for the Random Binary block and the Random Analog block. The panels differ only in their default values.



Attribute	Description
Min Value	The minimum output value for the block.
Max Value	The maximum output value for the block.
Sample Period	How often the block passes a new value.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*

For more information on...	See...	In this book...
Specifying Initial Values	page 83	*User's Guide*
Controlling the Flow of Data in an Application	page 32	*User's Guide*
Reading the Output Value	page 31	*Reference Manual*
Specifying How Often to Generate Values	page 31	*Reference Manual*

3

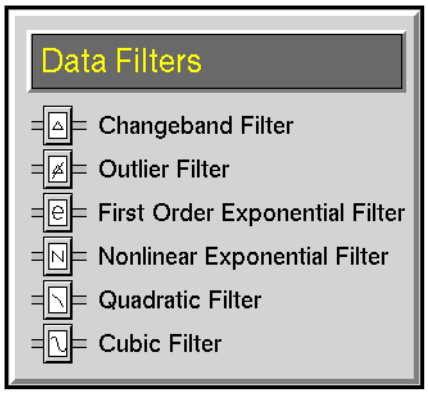
Data Filters

Describes the blocks used to filter data in a diagram, before analyzing it.

Introduction

GDA has several blocks that are specially designed for filtering data. These blocks are used after data entry blocks to filter out noise and find trends in data.

You can find the data filters on the Data Filters palette under the Data submenu of the Palettes menu:



Specifying How to Round Output Values

To have a filter round its output values, set the attribute Quantization. The block rounds its output value to the unit you specify. For example, if Quantization is 0.1, the block rounds to the nearest tenth, and if Quantization is 1.0, the block rounds to the nearest integer. All blocks on the Data Filters palette have this attribute. This table shows some examples of rounding:

If Quantization is…	The filter passes these values…		
none	0.5	1.43	1.77
0.1	0.5	1.4	1.8
0.5	0.5	1.5	2.0
1.0	1.0	1.0	2.0

This attribute is especially useful when you need to keep output values within known accuracy limits.

Filtering Values in a Range

The Changeband Filter and Outlier Filter help you with signals that need to stay within a range. You can specify whether the range remains constant or changes each time the filter passes a value. These are data reducing filters: they pass unmodified the values that satisfy a condition.

- The Changeband Filter block on page 67 passes a value only if the value is in a different band from the previous value. The possible bands are inside the range, above it, and below it. This filter is especially useful for ignoring insignificant changes.
- The Outlier Filter block on page 71 passes only those values that stay inside the range. This filter is useful for ignoring large sampling errors and impossibly extreme sensor readings.

Reducing Noise

The First-Order Exponential Filter and the Non-Linear Exponential Filter perform low-pass filtering and are useful for noise reduction. You can set how much filtering they perform with the attribute Filter Constant. These filters always pass values and modify them to reduce noise.

- The First-Order Exponential Filter block on page 75 passes large changes slowly, but its input values do not need to be equally spaced in time.
- The Non-Linear Exponential Filter block on page 78 passes large changes quickly, but its input values must be equally spaced in time.

Fitting to a Least-Squares Curve

The Quadratic Filter and Cubic Filter perform least-squares smoothing with a modified Savitsky-Golay filter. You can specify how many points these blocks operate on.

- The Quadratic Filter page 81 performs least-squares quadratic smoothing.
- The Cubic Filter page 81 performs least-squares cubic smoothing.

See Also

Many blocks that are not on the Filters palette are useful as simple filters. This section describes what those blocks are and where to find them.

Filtering Belief Values

Condition blocks can perform a type of filtering on inference values that is called hysteresis. For more information, see “Specifying Hysteresis” on page 103in the *GDA User’s Guide*.

Reducing Data

The Sample and Hold block on page 134 discards a certain number of inputs. You specify how many values it passes and discards. For example, you may have it discard every other value, or have it pass five values, discard ten, pass five, discard ten, and so on. This block is useful when you need to reduce the amount of data your application is processing or when you are using a block that operates on a fixed number of inputs, such as the Quadratic or Cubic Filters. Sample and Hold is on the Data Control palette.

Performing Statistical Operations

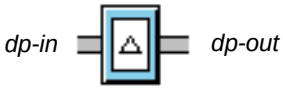
The Time Series palette contains several blocks you can use as filters. They perform statistical operations over the last several input values. You can specify how many values they operate on.

- The Moving Average block on page 165 computes the average.
- The Sample Median block on page 170 computes the median.
- The Variance block on page 159 computes the variance.
- The Discrete Rate of Change block on page 172 computes the rate of change or the slope.
- The Moving Range block on page 167 passes three values from its history: the maximum, minimum, and the difference between the maximum and minimum.

Fitting a Line to the Input

The Linear Fit block on page 156 finds the line that best fits the last several input values. You can specify how many values it operates on. It outputs two numbers: the slope of the line (which is the rate of change of the values) and the point on the line that fits the current input best. This block is on the Time Series palette.

Changeband Filter



The Changeband Filter passes a value only if it changes bands (that is, if the input value is in a different band from the previous value). There are three bands of values:

- Values within a range you specify.
- Values above that range.
- Values below that range.

Specify the range with the attributes Band Center and Band Range, described below.

The range can be fixed or floating. If Band Type is floating, each time a value changes bands, that value becomes the center of the range. If Band Type is fixed, the band always remains the same.

This filter is especially useful for data compression because it ignores small changes in input values. It is also useful with noisy sensors, because it passes only those values that differ significantly from previous values.

Note The Outlier Filter block on page 71 is similar to the Changeband Filter. The Outlier Filter passes values only if they stay within a band.

Specifying a Range

The attribute Band Range specifies the size of the block's range. Set it to the difference between the highest and lowest values in the range.

The attribute Band Type controls how the center of the range is set. If Band Type is fixed, you set the attribute Band Center and GDA uses that as the center. If Band Type is floating, GDA sets the center to the value passed whenever a value changes bands. GDA ignores the value of the attribute Band Center when Band Type is floating.

This table shows how the Band Range and the range's center describe the filter's bands:

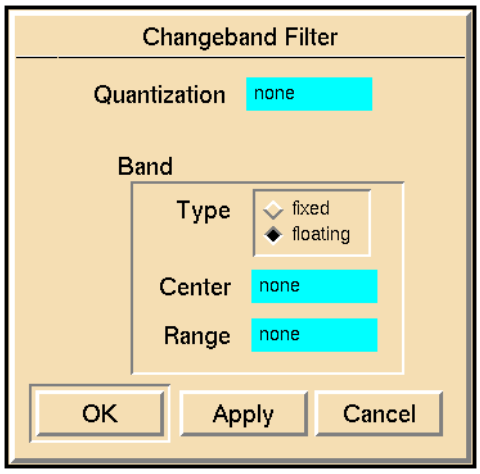
The band…	Contains the values…
Above the range	Greater than the center + Band Range/2
Within the range	Between the center – Band Range/2 and the center + Band Range/2
Below the range	Less than the center – Band Range/2

Usually, you set Band Type to floating, so the block passes only those values that differ significantly from previous values.

Note If Band Range is none, GDA uses a Band Range of 0.0. If Band Center is none, GDA proceeds as if Band Type were floating.

Configuring

This is the configuration panel for the Changeband Filter.



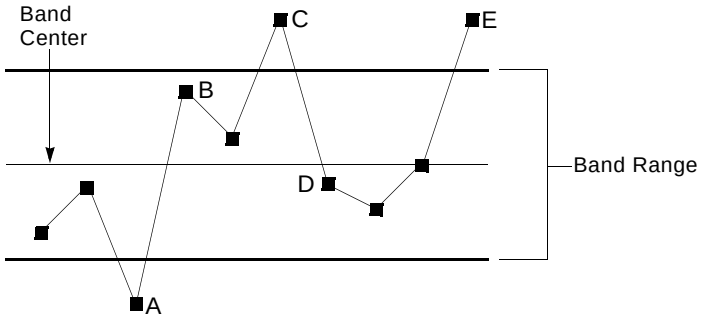
Attribute	Description
Quantization	A number that specifies the smallest increment that the block uses when rounding output values. For more information, see "Specifying How to Round Output Values" on page 64.
Band Type	How the center of the range is set. The values are either fixed, in which case the block uses the value of Band Center as the center, or floating, in which case the current output value is the new band center.
Band Center	The center of the band when Band Type is fixed; otherwise, this attribute is ignored.
Band Range	The extent of the band's range.

Examples

This section has some examples of a fixed and floating Changeband Filter.

Fixed Band

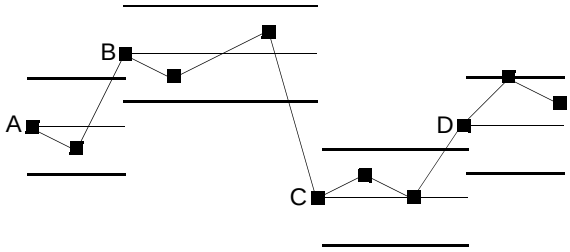
This figure shows some input values for a Changeband Filter with Band Type set to fixed. The Band Range and Band Center are marked. The filter passes only the values marked with a letter.



A is the first new value outside the range, so it is passed. B is the first new value to re-enter the range. C is the first new value outside the range. D is the first new value to re-enter the range again. And E is the first new value outside the range.

Floating Band

This figure shows some input values for a Changeband Filter with Band Type set to fixed. The range is marked with bold lines. The center of the range is marked with a thin line. The filter passes only the values marked with a letter.



A is the original center. B is the first new value outside A's range. C is the first new value outside B's range. And D is the first new value outside C's range.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Round Output Values	page 64	*Reference Manual*
The Moving Range block	page 167	*Reference Manual*

Outlier Filter

The Outlier Filter passes only values that are within a band you specify. Specify the band with the attributes Band Center and Band Range. If Outlier Replacement is yes, the filter replaces out-of-range values with the nearest in-range value. If Outlier Replacement is no, the filter discards out-of-range values.

The band can be fixed or floating. If Band Type is floating, each time the block passes a value, that value becomes the center of the range. If Band Type is fixed, the range always remains the same.

Note The Changeband Filter block on page 67 is similar to the Outlier Filter. The Changeband Filter passes values only if they change bands.

Specifying a Range

The attribute Band Range specifies the size of the block's range. Set it to the difference between the highest and lowest values in the range. The filter's range contains the values between the range's center minus Band Range/2 and the range's center plus Band Range/2.

The attribute Band Type controls how the range's center is set. If Band Type is fixed, you set the attribute Band Center and GDA uses the value of Band Center as the center of the range.

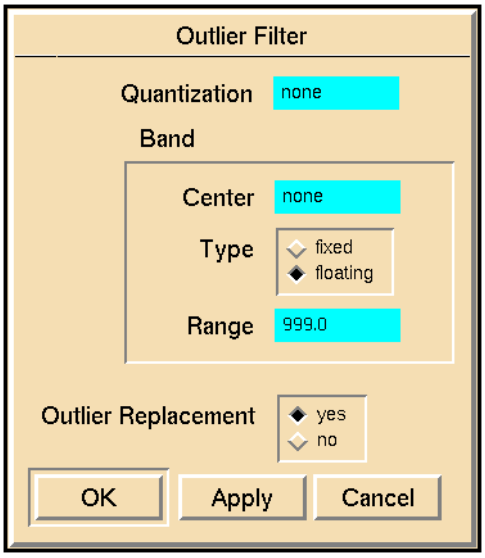
If Band Type is floating, GDA uses the last value it passed as the center, and GDA ignores the value of the attribute Band Center. Because the block always passes in-range values, the center is always set to an in-range input value.

When the block receives an out-of-range value, what it passes depends on the value of Outlier Replacement, the attribute which controls whether the block replaces an out-of-range value with another value. If Outlier Replacement is yes, it changes the center of the range to the in-range value nearest to the input value. If Outlier Replacement is no, the center does not change.

Note If Band Range is none, GDA uses a Band Range of 0.0. If Band Center is none, GDA proceeds as if Band Type were floating.

Configuring

This is the configuration panel for the Outlier Filter.



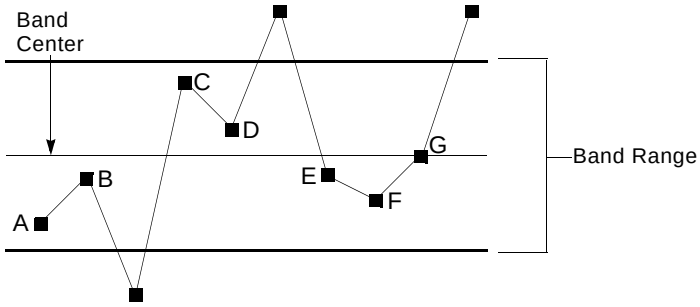
Attribute	Description
Quantization	A number that specifies the smallest increment that the block uses when rounding output values. For more information, see “Specifying How to Round Output Values” on page 64.
Band Center	The center of the band when Band Type is fixed; otherwise, this attribute is ignored.
Band Type	How the center of the range is set. The values are either fixed, in which case the block uses the value of Band Center as the center, or floating, in which case the current output value is the new band center.
Band Range	The extent of the band’s range.
Outlier Replacement	When Band Type is floating, whether the block replaces the band center with the nearest in-range value when the block receives an out-of-range value (yes), or whether the center remains the same (no).

Examples

This section has some examples of a fixed and floating Outlier Filter.

Fixed Band

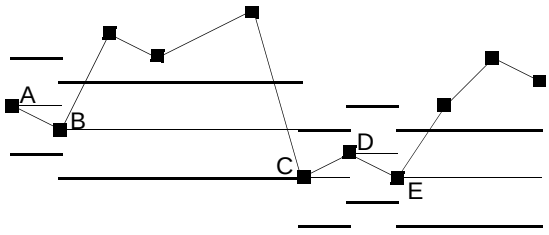
This figure shows some input values for an Outlier Filter with Band Type set to fixed. The Band Range and Band Center are marked. Letters mark values that are passed without replacement. If Outlier Replacement is yes, the filter replaces unmarked values with the nearest in-range value. Otherwise, the filter does not pass any value.



Floating Band

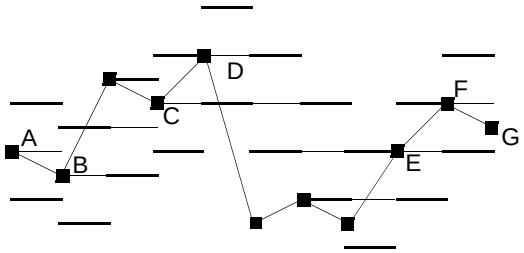
The next two figures show some input values for an Outlier Filter with Band Type set to floating. They differ in how Outlier Replacement is set. In both, the range is marked with a bold line, and the center of the range is marked with a thin line.

In this figure, Outlier Replacement is no. Notice that the center of the range changes only when a new input value falls in the same range as the previous value. Letters mark values that the block passes.



In this figure, Outlier Replacement is yes. Notice that when a value is out of range, the center of the new range becomes the maximum or minimum of the old

range. Letters mark values that the block passes without replacing. For unmarked values, the filter passes the closest in-range value.

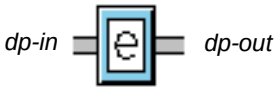


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Round Output Values	page 64	*Reference Manual*
The Moving Range block	page 167	*Reference Manual*

First-Order Exponential Filter



The First Order Exponential Filter performs low-pass filtering to filter out high frequency noise. Its output value depends on the previous output value and the current input value. To specify how much the filter weights previous output values, set the attribute Filter Constant. The larger the Filter Constant, the more weight the previous output values have.

This filter does not require inputs that are equally spaced in time.

The Non-Linear Exponential Filter block on page 78 responds more quickly to large changes, but its input must arrive at equal time intervals.

Filtering

The First-Order Exponential Filter uses this equation to compute its output value. The variables in the equation are explained in the table that follows.

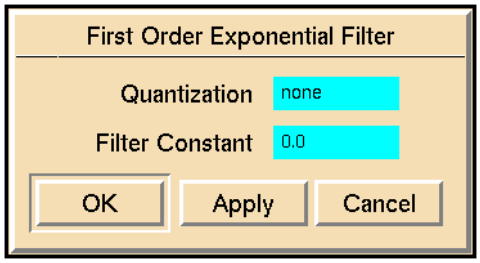
$$\text{output}_n = \alpha \cdot \text{input} + (1-\alpha)\text{output}_{n-1}, \text{ where } \alpha = e^{-\frac{\Delta\text{time}}{\textit{Filter-constant}}}$$

This variable…	Is…
output_n	The current output value
output_{n-1}	The previous output value
input	The current input value
Δtime	The difference in time between the arrival of the last input value and the current input value

The coefficient α is the amount of weight that the filter gives the current input, and the coefficient $(1-\alpha)$ is the amount of weight that the filter gives the previous output value. The block computes α with both the Filter Constant and the difference in time between the arrival of the last input and the arrival of the current input. The constant α is computed such that values received long ago will not have as much effect on the current value as ones that were received a short time ago.

Configuring

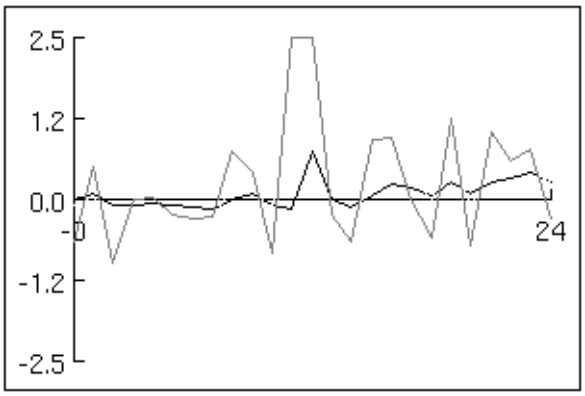
This is the configuration panel for the First Order Exponential Filter.



Attribute	Description
Quantization	A number that specifies the smallest increment that the block uses when rounding output values. For more information, see “Specifying How to Round Output Values” on page 64.
Filter Constant	The amount of weight that the filter gives the current input over the previous input.

Example

This figure shows the chart of a First-Order Exponential Filter with a Filter Constant of 5. The raw signal has more variation and the filtered signal is smoother.

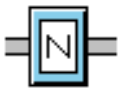


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Round Output Values	page 64	*Reference Manual*
Changeband Filter	page 67	*Reference Manual*
Moving Average	page 165	*Reference Manual*

Non-Linear Exponential Filter

dp-in  *dp-out*

The Non-Linear Exponential Filter performs low-pass filtering to filter out high frequency noise. It responds immediately to large changes in its input values. The Non-Linear Exponential Filter is especially useful for an alarm that needs to filter noisy input but also needs to respond quickly to drastic changes.

To specify how much weight to give to the current input and the previous output, set the attributes Filter Constant and the Noise Std Deviation. This filter behaves like a First-Order Exponential Filter with a Filter Constant that varies depending on how much the filtered values and raw measurements differ.

To specify how much weight the filter gives previous output values, set the attribute Filter Constant. The larger the Filter Constant, the more weight the previous output values have. Set it to a value between 3.0 and 5.0.

To specify the common range of values for the filter's input, set the Noise Std Deviation to an estimate of the standard deviation of the noise level. Usually, changing the Noise Std Deviation just changes the weight of the current input value. But when the difference between the current input value and the previous output value is greater than (Filter Constant * Noise Std Deviation), the filter passes the current input value unchanged.

Note This filter's inputs must be spaced equally in time. Avoid using it with data compression blocks like the Changeband and Outlier Filters.

You can use the First-Order Exponential Filter block on page 75 with inputs that are not equally spaced in time. However, it responds slowly to large changes in the input values.

Filtering

The Non-Linear Exponential Filter uses this equation to compute its output value. The variables in the equation are explained in the table that follows.

$$\text{output}_n = P \times \text{input}_n + (1 - P)\text{output}_{n-1}$$

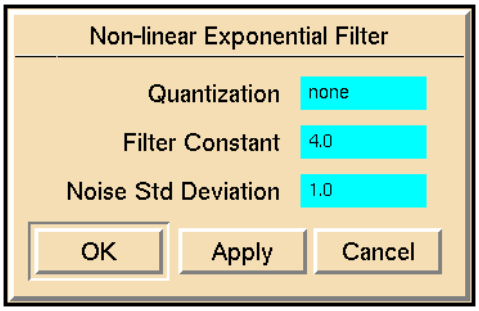
$$\textit{where}\ \ P = \begin{pmatrix} 1 - \left|\dfrac{\Delta\text{input}}{FC \times NSD}\right|, & \text{if } \Delta\text{input} < FC \times NSD \\ 0, & \text{if } \Delta\text{input} > FC \times NSD \end{pmatrix}$$

This variable…	Is…
$output_n$	The current output value
$output_{n-1}$	The previous output value
input	The current input value
Δinput	The difference between the current input value and the previous input value
FC	The attribute Filter constant
NSD	The attribute Noise Std Deviation

Notice that when the difference in the current input value increases by more than (Filter Constant * Noise Std Deviation), the filter passes the current input unchanged. Also, note that time is not a factor in the equation, so the filter's inputs must be spaced equally in time to be accurate.

Configuring

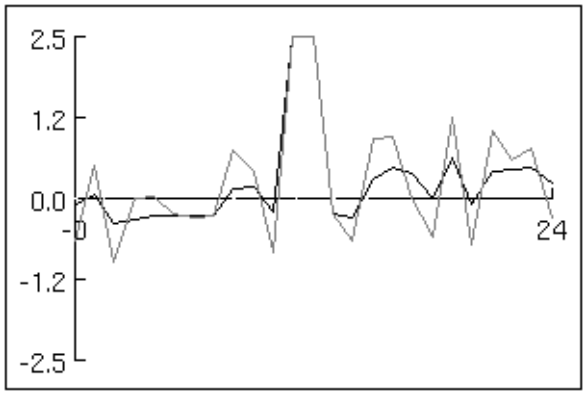
This is the configuration panel for the Non-Linear Exponential Filter.



Attribute	Description
Quantization	A number that specifies the smallest increment that the block uses when rounding output values. For more information, see "Specifying How to Round Output Values" on page 64.
Filter Constant	The amount of weight that the filter gives the previous input over the current input.
Noise Std Deviation	An estimate of the standard deviation of the noise in the signal.

Example

This figure shows a Non-Linear Exponential Filter with a Filter Constant of 5. The raw signal is in gray and the filtered signal is in black.

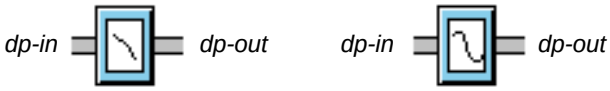


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Round Output Values	page 64	*Reference Manual*

Quadratic Filter, Cubic Filter



The Quadratic Filter and Cubic Filter use a modified Savitsky-Golay algorithm to perform low-pass filtering. The Quadratic Filter performs least-squares quadratic smoothing. The Cubic Filter performs least-squares cubic smoothing.

Note These filters' inputs must be spaced equally in time. Avoid using them after data compression blocks like the Changeband and Outlier Filters. If you need to reduce the sample rate of your data, use them after a Sample and Hold block on page 134.

Specifying the Number of Points

To specify how many points the block uses to perform its filtering, set the attribute Sample Size to either 5 or 7. The block does not start passing values until it receives at least Sample Size input values. If Sample Size is 5, it uses a 5-point filter. If Sample Size is 7, it uses a 7-point filter.

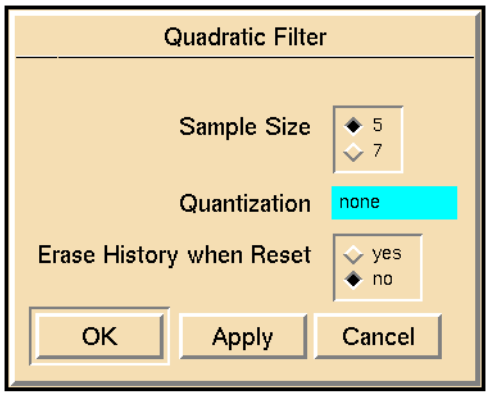
Note Unlike most blocks that maintain a history, the Quadratic Filter and Cubic Filter do not have the attribute Sample Type and limit what the Sample Size can be. Sample Size must be either 5 or 7 points.

Resetting

You can determine what happens to the history when the block is reset. If the attribute Erase History When Reset is yes, the block erases the history and it will not pass a value again until it receives at least Sample Size new points. If Erase History When Reset is no, it keeps the history and continues passing values right away.

Configuring

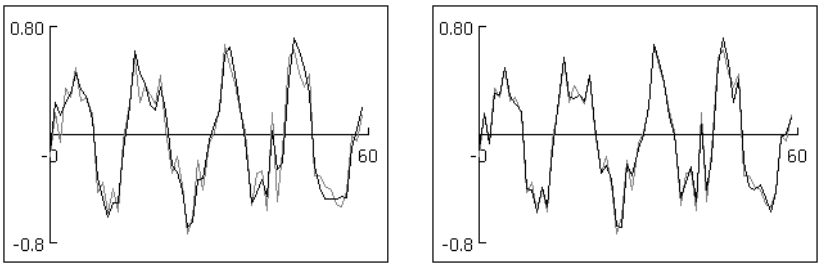
This is the configuration panel for the Quadratic Filter. The panel for the Cubic Filter is identical except for the block name.



Attribute	Description
Sample Size	The number of points the block uses before performing its filtering.
Quantization	A number that specifies the smallest increment that the block uses when rounding output values. For more information, see “Specifying How to Round Output Values” on page 64.
Erase History When Reset	Whether the block erases its history when reset (yes) or continues passing values based on the current history after reset (no).

Example

This figure shows charts of a Quadratic Filter and a Cubic Filter, both with a Sample Size of 7. (The Quadratic Filter is first and the Cubic Filter is second.) The raw signal is in gray and the filtered signal is in black.



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Round Output Values	page 64	*Reference Manual*
The Discrete Rate of Change block	page 172	*Reference Manual*

4

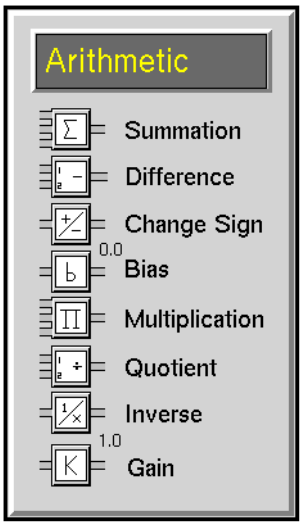
Arithmetic

Describes the blocks that perform arithmetic operations on numeric data.

Introduction

GDA provides you with several blocks that let you perform arithmetic operations on data values: addition, subtraction, multiplication, and division.

You can find the Arithmetic palette under the Data submenu of the Palettes menu:



Adding and Subtracting Data

These blocks let you add and subtract data:

- The Summation block on page 88 adds any number of input values.
- The Difference block on page 90 subtracts one value from another.
- The Change Sign block on page 91 inverts the sign of a values.
- The Bias block on page 92 adds a constant value to its input.

Multiplying and Dividing Data

These blocks let you multiply and divide data:

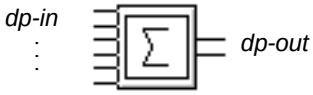
- The Multiplication block on page 94 multiplies any number of input values.
- The Quotient block on page 96 divides one input value by another.
- The Inverse block on page 97 passes the inverse of a value.
- The Gain block on page 98 multiplies its input by a constant value.

See Also

Other blocks also perform mathematical operations:

- When you use fuzzy logic, many of the blocks on the Logic Gates palette in the Inference Blocks menu perform mathematical operations on belief values. The blocks on the Evidence Gates palette in the Inference Blocks menu also perform mathematical operations on belief values.
- To perform statistical operations on data values, such as computing the average, median, minimum, and maximum value, use the blocks on the Functions palette and Time Series palette in the Data Blocks menu.

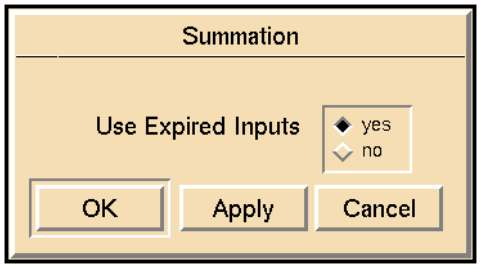
Summation



The Summation block adds all its input values together and passes the result. This block is a peer input block; you can add inputs by dragging additional input paths into the block.

Configuring

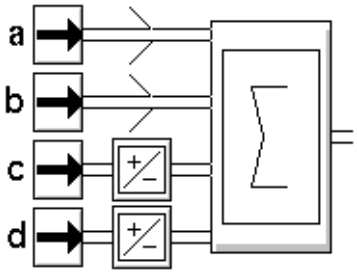
This is the configuration panel for the Summation block.



Attribute	Description
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

Example

This figure shows a diagram that computes (a + b) - (c + d). Note that the Summation block has an extra input port and has been enlarged with the Change Size menu choice.

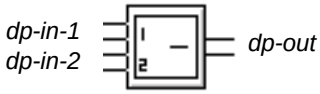


See Also

For more information on how to use this block, see the pages below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*

Difference



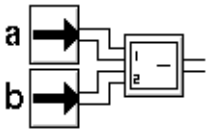
The Difference block subtracts the value of the lower path from the value of the upper path and passes the result.

Configuring

The Difference block has no configurable attributes.

Example

This figure shows the equation a - b.

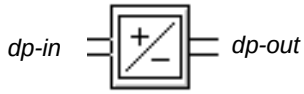


See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Change Sign



The Change Sign block changes the sign of the input value, as if you multiplied it by –1. This block makes negative values positive and positive values negative.

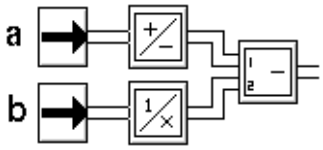
Configuring

The Change Sign block has no configurable attributes.

Example

This figure shows a diagram that computes this equation:

$$-a-\frac{1}{b}.$$



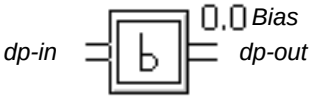
The block marked "1/x" is an Inverse block on page 97. The block marked "–" is a Difference block on page 90.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

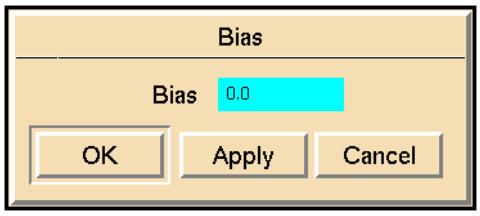
Bias



The Bias block adds a fixed constant to its input value. Specify the constant in the block's Bias attribute.

Configuring

This is the configuration panel for the Bias block.



Attribute	Description
Bias	The value that the block adds to its input.

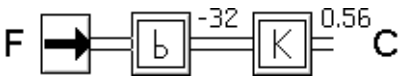
Example

This figure shows the equation for converting temperatures from Fahrenheit to Celsius:

$$°C = \frac{°F - 32}{1.8}$$

or

$$°C = (°F + (-32)) \times 0.56$$



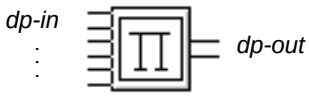
The block marked "K" is the Gain block on page 98.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*

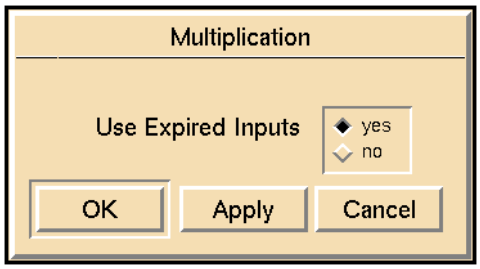
Multiplication



The Multiplication block multiplies all its input values and passes the result.

Configuring

This is the configuration panel for the Multiplication block.



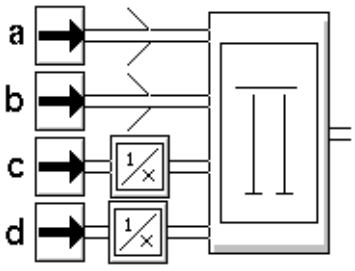
Attribute	Description
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

Example

This figure shows a diagram that computes this equation:

$$\frac{ab}{cd}$$

The Multiplication block has an extra input port and has been enlarged with the Change Size menu choice.

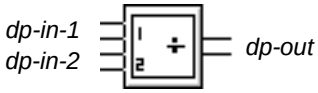


See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*

Quotient



The Quotient block divides the value of the upper path by the value of the lower path and passes the result.

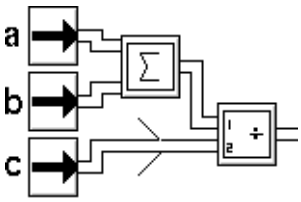
Configuring

The Quotient block has no configurable attributes.

Example

This figure shows this equation:

$$\frac{a + b}{c}$$

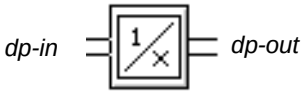


See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Inverse



The Inverse block computes the multiplicative inverse of the input value.

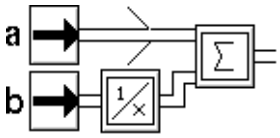
Configuring

The Inverse block has no configurable attributes.

Example

This figure shows a diagram that computes this equation:

$$a + \frac{1}{b}$$



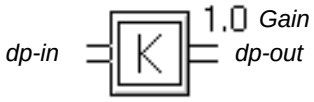
The block marked "Σ" is a Summation block on page 88.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

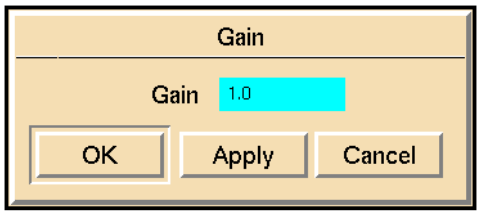
Gain



The Gain block multiplies its input value by a fixed constant. Specify the constant in the block's Gain attribute.

Configuring

This is the configuration panel for the Gain block.

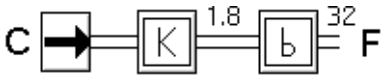


Attribute	Description
Gain	The value that the block multiplies by its input.

Example

This figure shows the equation for converting temperatures from Celsius to Fahrenheit using this equation:

$°F = °C \times 1.8 + 32$



The block marked "b" is the Bias block on page 92.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

For more information on...	See...	In this book...
Editing Attribute Displays	page 27	*User's Guide*
The Belief Weight block	page 322	*Reference Manual*

5

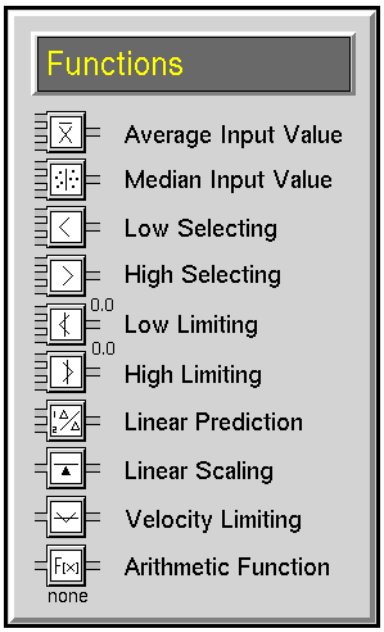
Functions

Describes the blocks that perform statistical operations on data values.

Introduction

The Functions palette contains several blocks that perform statistical operations on data values: computing the average, median, minimum, or maximum value for any number of input values. It also includes blocks to convert values from one linear scale to another and predict future values for points on a line. Finally, it has a block that lets you define your own function to apply to data values.

You can find the Functions palette under the Data submenu of the Palettes menu:



Averaging Values

These blocks compute the average or median value of their input values:

- The Average Input Value block on page 104 computes the average value of any number of input values.
- The Median Input Value block on page 106 computes the median value of any number of input values.

Computing the Minimum or Maximum

These blocks compute the minimum or maximum value of their input values:

- The Low Selecting block page 108 passes on the minimum value of its input values.
- The High Selecting block page 108 passes on the maximum value of its input values.
- The Low Limiting block page 110 passes on either the minimum value of its input values or a limit if the lowest input value is less than that limit.
- The High Limiting block page 110 passes on either the maximum value of its input values or a limit if the highest input value is greater than that limit.

Predicting Values

The Linear Prediction block on page 113 predicts a future value for a point on a line, given a point on the line and the slope of the line. It is related to the Linear Fit block on page 156.

Converting Values

The Linear Scaling block on page 115 converts its input value from one scale to another scale, such as from Fahrenheit to Celsius. It is useful for converting data values to belief values.

Spreading Out Change

The Velocity Limiting block on page 119 damps large changes in its input value by spreading them out over time.

Defining Your Own Function

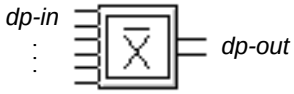
The Arithmetic Function block on page 122 lets you apply your own function or a built-in G2 function to its input value.

See Also

Here are some other blocks that also perform mathematical operations:

- To perform mathematical operations on belief values, use the blocks on the Logic Gates palette in the Inference Blocks menu. These blocks are useful in fuzzy logic computations. For example, the AND Gate passes the minimum of its input belief values and the OR Gate passes the maximum of its input belief values.
- To perform statistical operations on a history of data values, use the Time Series palette on the Data Blocks menu.

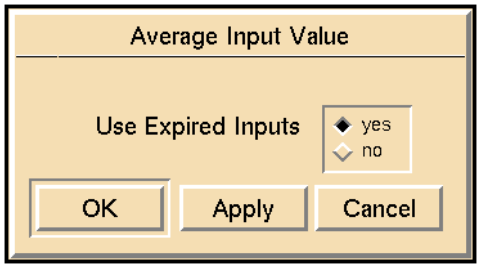
Average Input Value



The Average Input Value block passes the average of all its input values. It accepts any number of inputs.

Configuring

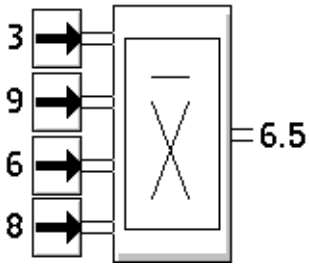
This is the configuration panel for the Average Input Value block.



Attribute	Description
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

Example

This figure shows an Average Input Value block with the input values 3, 9, 6, and 8. It passes the value 6.5. Note that the Average Input Value block has an extra input port and has been enlarged with the Change Size menu choice.

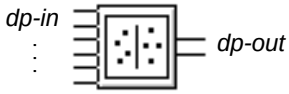


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
The Moving Average block	page 165	*Reference Manual*

Median Input Value

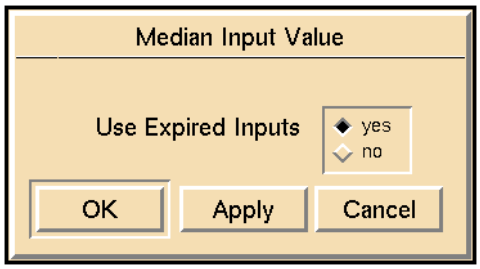


The Median Input Value block passes the median of all its input values. It accepts any number of inputs.

If the Median Input Value block has an odd number of input values, it passes the middle input value. If the block has an even number of input values, it passes the minimum of the two middle input values.

Configuring

This is the configuration panel for the Median Input Value block.

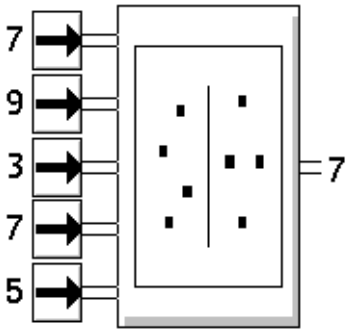


Attribute	Description
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

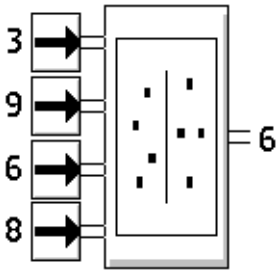
Example

The following figures are examples of how a Median Input Value block handles both odd and even numbers of inputs. In both cases, note that the Median Input Value block has extra input ports and has been enlarged with the Change Size menu choice.

This figure shows a Median Input Value block connected to five inputs. It passes 7. (The median of 3, 5, 7, 7, 9 is the middle value.)



This figure shows a Median Input Value block connected to four inputs. It passes 6. (The median of 3, 6, 8, 9 is the lesser of 6 and 8.)

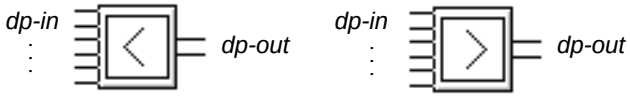


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
The Sample Median block	page 170	*Reference Manual*

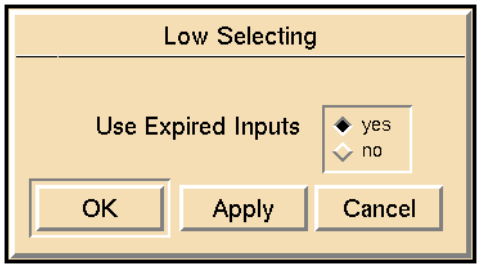
Low Selecting, High Selecting



The Low Selecting and High Selecting blocks choose the minimum or maximum value of their input values. They accept any number of inputs. The High Selecting block passes the maximum input value. The Low Selecting block passes the minimum input value.

Configuring

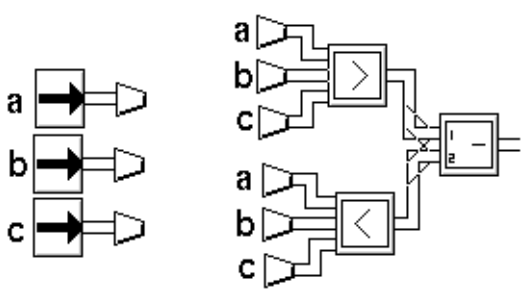
This is the configuration panel for the Low Selecting block. The panel for the High Selecting block is identical except for the block name.



Attribute	Description
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

Example

This figure calculates the difference between the maximum and minimum of three values:

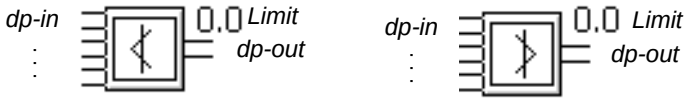


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*

Low Limiting, High Limiting



The Low Limiting and High Limiting blocks choose the minimum or maximum value from a group of values, but let you put a limit on the values they pass. If the minimum or maximum value does not exceed the limit, the blocks pass the minimum or maximum. If the minimum or maximum value exceeds the limit, the blocks pass the limit instead. You set the limit with the Limit attribute. These blocks can take any number of inputs.

The High Limiting block passes one of these values:

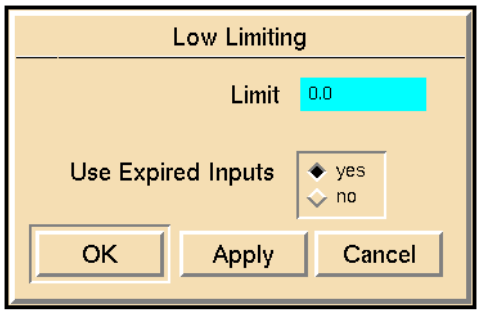
- If the maximum input value is less than or equal to Limit, the maximum value.
- If the maximum input value is greater than Limit, Limit.

The Low Limiting block passes one of these values:

- If the minimum input value is greater than or equal to Limit, the minimum value.
- If the minimum input value is less than Limit, Limit.

Configuring

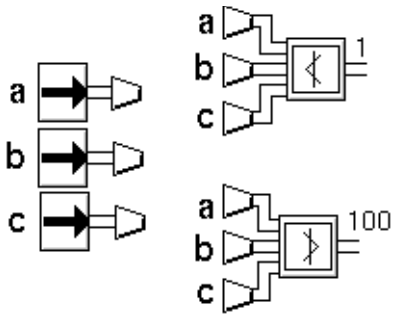
This is the configuration panel for the Low Limiting block. The panel for the High Limiting block is identical except for the block name.



Attribute	Description
Limit	The maximum or minimum value, depending on the block.
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

Example

This figure shows a High Limiting and a Low Limiting block connected to three entry points:



This table shows what values the blocks in the previous figure would pass for certain input values. The High Limiting block has a Limit of 100. The Low Limiting block has a Limit of 1.

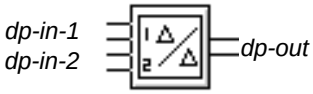
If the blocks receive...	Low Limiting passes…	High Limiting passes…
23, 83, 46	23	83
71, -6, 38	1	71
62, 17, 115	17	100

See Also

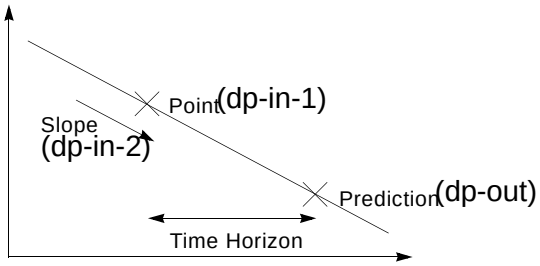
For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*

Linear Prediction



The Linear Prediction block predicts a future value for a point on a line, given another point on the line and the line's slope. Pass the given point into the top input port, (dp-in-1), and pass the slope into the bottom input port (dp-in-2). Specify the amount of time from the given point to the predicted point with the attribute Time Horizon. This figure shows what these values represent:

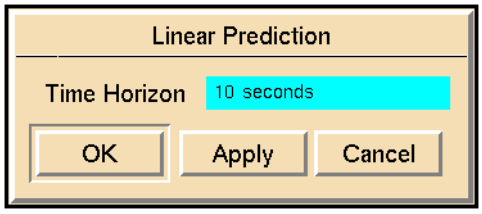


This block is especially useful when you need to respond to an event before it happens. For example, you may be monitoring a valve which takes a minute to close once you have started closing it. You could use a Linear Prediction block to predict whether you will need it closed in a minute, and start closing it in advance. Or you may be monitoring inventory in a stock room, and a shipment arrives one week after you have ordered it. You could use a Linear Prediction block to predict whether you will need more inventory in a week, and order it in advance.

This block is frequently used with a Linear Fit block on page 156 which fits a history of values to a line. The Linear Prediction block's input values are the same as the Linear Fit block's output values: a point on a line and the line's slope, so you can connect a Linear Fit block directly to a Linear Prediction block.

Configuring

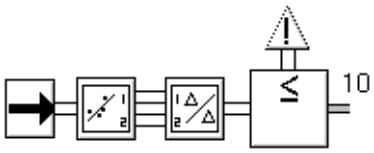
This is the configuration panel for the Linear Prediction block.



Attribute	Description
Time Horizon	The amount of time between the point given as an input and the predicted point.

Example

You may want a diagram that raises an alarm if there will be ten or fewer widgets in the supply room next week, so you can order more. The following diagram receives input from an entry point, fits those values to a line using a Linear Fit block with a Time Horizon of 1 week, predicts a future value with a Linear Prediction block, and raises an alarm if that future value is 10 or less.

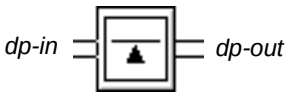


See Also

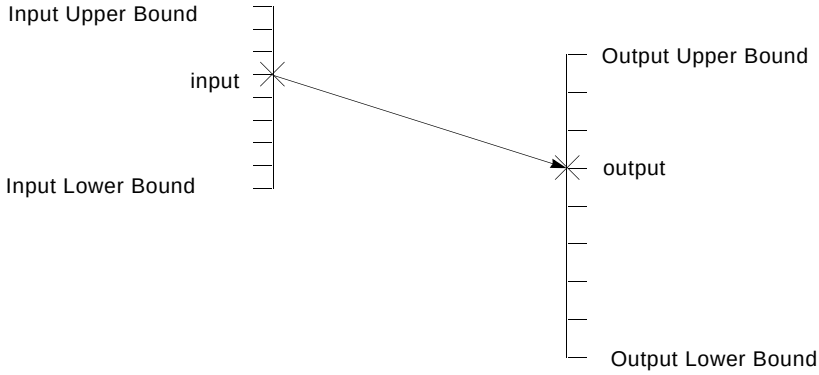
For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Linear Fit block	page 156	*Reference Manual*

Linear Scaling



The Linear Scaling block scales the input data value from one range to another, as in this figure. For example, you can use it to convert temperatures from Fahrenheit (32 to 212) to Celsius (0 to 100) or to convert data values to belief values (0 to 1).



Specify the range of the input data value with the attributes Input Upper Bound and Input Lower Bound. Specify the range of the output data value with the attributes Output Upper Bound and Output Lower Bound. The block computes the output value with this equation. The variables in the equation are explained in the table that follows.

$$\text{output} = O_L + \frac{\text{input} - I_L}{I_U - I_L}(O_U - O_L)$$

This variable…	Is…
output	The current output value
input	The current input value
IU	The attribute Input Upper Bound
IL	The attribute Input Lower Bound

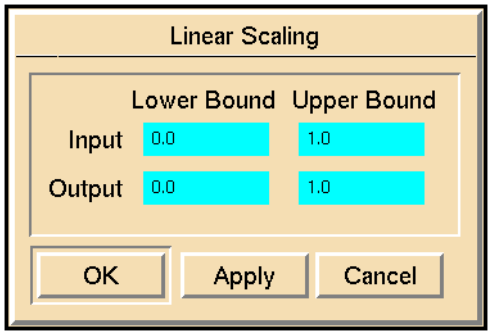
This variable…	Is…
OU	The attribute Output Upper Bound
OL	The attribute Output Lower Bound

If you pass the block any value greater than the Input Upper Bound, it will pass the Output Upper Bound. And if you pass it any value less than the Input Lower Bound, it will pass the Output Lower Bound.

Note The Input Lower Bound cannot equal the Input Upper Bound. If they are equal, the block will try to divide by zero and produce an error.

Configuring

This is the configuration panel for the Linear Scaling block.



Attribute	Description
Input Lower Bound	The lower range of the input value.
Input Upper Bound	The upper range of the input value.
Output Lower Bound	The lower range of the output value.
Output Upper Bound	The upper range of output value.

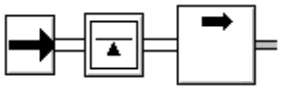
Example

To convert temperatures from Fahrenheit to Celsius, set up a Linear Scaling block with the attributes in this table:

Set this attribute…	To this value...
Input Upper Bound	212
Input Lower Bound	32
Output Upper Bound	100
Output Lower Bound	0

Note that for any temperature below 32°F, this block will pass O°C, and for any temperature above 212°F, it will pass 100°C. If you want this block to handle a larger range, you must set the bounds differently.

Suppose you want to convert a data value into a belief value. You know that if a tank's temperature is 50°, it is definitely working properly, and if its temperature is 100°, it is definitely broken. For temperatures between 50 and 100, you want to produce belief values that are scaled accordingly. Set up a diagram with a Linear Scaling block and a Belief Input block on page 253, as in the following figure, and set up the Linear Scaling blocks attributes as in the following table.



Set this attribute…	To this value...
Input Upper Bound	100
Input Lower Bound	50
Output Upper Bound	0.0
Output Lower Bound	1.0

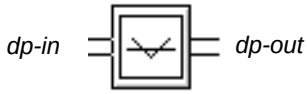
Note that for any temperature below 50°, this block passes 0 (completely false), and for any temperature above 100°, it passes 1 (completely true). In this example, that is the behavior you want.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Velocity Limiting



The Velocity Limiting block damps large changes in the input value so that the change is spread out over time. You can specify how much the input value can change before it is damped and how quickly it is damped. Specify the maximum amount of increase of the input with Max Rise Velocity. Specify the maximum amount of decrease with Max Fall Velocity. Specify how quickly it is damped with Update Interval.

If the block receives an input value that changes by less than the amounts you specified, it passes on the value unchanged. However, when this block receives a value that is greater than the prior output + Max Rise Velocity (or less than prior output – Max Fall Velocity), it damps the change as follows:

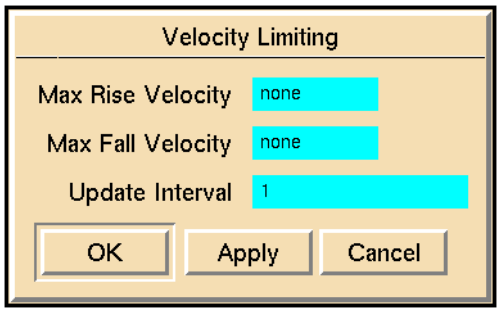
1. Immediately, it passes on the previous output value + Max Rise Velocity × Update Interval (or – Max Fall Velocity × Update Interval).
2. Every Update Interval, it increments the previous output value by Max Rise Velocity × Update Interval (or decrements it by Max Fall Velocity × Update Interval).
3. If the block receives a new input value, it starts from step 1 again.
4. When the output value equals the latest input value, the block stops changing the output value. It never goes above (or below) the latest input value. If necessary, it adds less than the Max Rise Velocity × Update Interval (or subtracts less than the Max Fall Velocity × Update Interval) to reach the latest input value.

This table summarizes how a Velocity Limiting block damps its output value:

If the block receives an input value…	It does this immediately and every Update Interval until the output equals the input…
Greater than prior output + Max Rise Velocity	Increments prior output by Max Rise Velocity × Update Interval
Less than prior output – Max Fall Velocity	Decrements prior output by Max Fall Velocity × Update Interval

Configuring

This is the configuration panel for the Velocity Limiting block.



Attribute	Description
Max Rise Velocity	The maximum amount of rise of the input.
Max Fall Velocity	The maximum amount of decrease of the input.
Update Interval	How quickly the input value is damped.

Example

Suppose you have a Velocity Limiting block that is set up like this:

Attribute	Value
Max Rise Velocity	1
Max Fall Velocity	1
Update Interval	1 second

Here are some samples of how this block damps its output value:

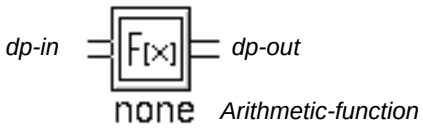
- If the block receives 1 and then 5, it passes on these values one per second: 1, 2, 3, 4, 5.
- If the block receives 1 and then 4.5, it passes on: 1, 2, 3, 4, 4.5.
- If the block receives 1 and 5, and then receives 3 before it passes 3 as output, it passes on: 1, 2, 3.
- If the block receives 5 and 10, and then receives 3 after it passes 7 as output, it passes on: 5, 6, 7, 6, 5, 4, 3.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Filters palette	page 63	*Reference Manual*

Arithmetic Function



The Arithmetic Function block lets you use a function or procedure as a block in a diagram. The block applies the function or procedure to its input value and passes the result. Specify the name of the function in the attribute Arithmetic Function.

You can use any:

- Built-in G2 function
- User-defined function
- Procedure
- Tabular-function

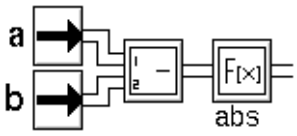
Note The input value for this block can be text, symbolic, or numeric.

Built-in G2 Function

You may set the attribute Arithmetic Function to any of these built-in G2 functions:

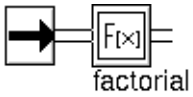
abs	arctan	ceiling	cos
exp	floor	int	ln
log	random	sin	sqrt
tan	truncate		

This figure shows a diagram that computes the absolute value of the difference of two values:



User-Defined Function

You may use any user-defined function that accepts one quantitative argument and returns one quantitative value. Set the attribute Arithmetic Function to the name of the function. This figure shows a diagram that figures the factorial of a value:



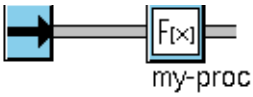

```
factorial (n) =
    (if n <= 1
     then 1
     else n * factorial(n - 1))
```

Procedure

You can use a procedure that accepts three arguments, described in the following table, and returns a single value. Set the attribute Arithmetic Function to the name of your procedure. The block passes the procedure's return value as its output value. The block automatically passes the Collection-time and Quality of the block's input data path.

Argument	Type	Description
input-value	value	The block's input value.
collection-time	float	The Collection-time of the block's input path.
quality	symbol	The Quality of the block's input path.

This figure shows a diagram with an Arithmetic Function block that uses a procedure:

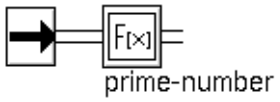



```
my-proc(input-value: float, collection-time:
  float, quality: symbol) = (float)
delta: float;
begin
   delta = the current time - collection-time;
   if delta > 10 then
      return 0
   else if delta > 5 then
      return input-value/2
   else
      return input-value
end
```

Tabular Function

You can use a tabular-function-of-1-arg that accepts one argument and returns a value. Set the attribute Arithmetic Function to the name of your function. The block passes its input value to the function as an argument and passes the function's return value as its output value. If the function cannot evaluate the input value, GDA signals an error. For information on a G2 tabular-function-of-1-arg, see the *G2 Reference Manual*.

This table shows a diagram with an Arithmetic Function block that uses a tabular-function-of-1-arg:



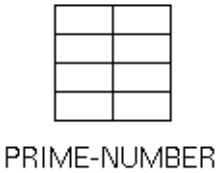
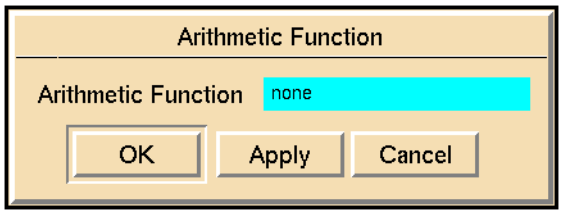


Table of values	
add or delete rows	
x	prime-number (x)
1	1
2	2
3	3
4	5
5	7

Configuring

This is the configuration panel for the Arithmetic Function block.



Attribute	Description
Arithmetic Function	The name of a G2 function that the block executes.

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Custom Block Wizard	page 169	*User's Guide*
The Generic Action block	page 463	*Reference Manual*
The Encapsulation Block	page 479	*Reference Manual*

6

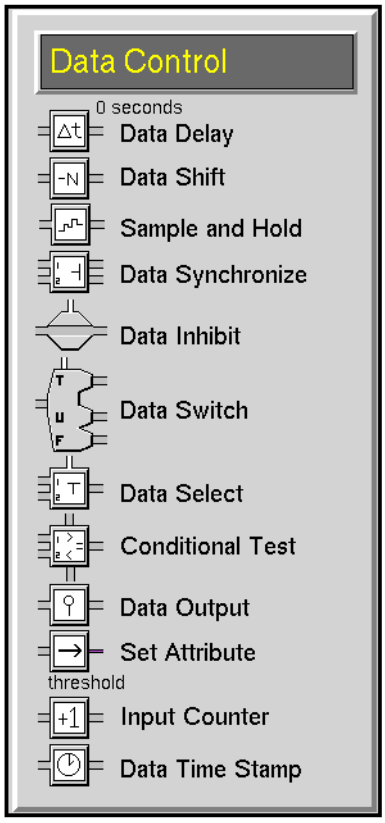
Data Control

Describes the blocks that control how data flows through your diagram.

Introduction

These blocks let you control how data flows through your diagram. You can stop the flow, divert it, combine paths together, and set G2 variables and GDA block attributes to the path's data value.

You can find the Data Control palette in the Data submenu of the Palettes menu:



Stopping and Pausing Paths

These blocks let you stop the flow of data or pause it until some condition is met.

- The Sample and Hold block on page 134 reduces the amount of data passed by selectively passing or discarding values.
- The Data Delay block on page 130 pauses the flow of data for a set period of time.
- The Data Shift block on page 132 delays passing its input value until it has received more input values.
- The Data Synchronize block on page 136 synchronizes two data paths by waiting for data from both paths to arrive before passing a value.

- The Data Inhibit block on page 138 holds an incoming data value as long as an inference input has a given value.

Combining and Splitting Paths

These blocks let you combine two data paths or split off one path into three different paths.

- The Data Select block on page 142 combines two paths together. It chooses which of the values to pass on by checking the value of an inference path.
- The Conditional Test block on page 143 splits one path into three. It chooses which path to send the input value to by comparing that input value against a second input value.
- The Data Switch block on page 140 splits one path into three. It chooses which path to send the input value to based on an inference value.

Outputting Data

These blocks let you set a block's attribute, G2 parameter, or G2 variable to the input data value.

- The Set Attribute block on page 147 sets another block's attribute to the input data value.
- The Data Output block on page 145 sets a G2 parameter or G2 variable to the input data value.

Passing the Data Count or Time Stamp

These blocks pass along the number of data values they have received or the time they received their latest values.

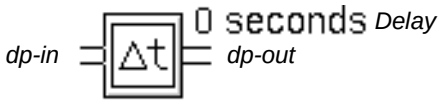
- The Input Counter block on page 150 passes the number of times it received a data value.
- The Data Time Stamp block on page 152 passes the time stamp of the last value it received.

See Also

GDA has many other blocks that control the flow of information, but that take inference values or control signals as input:

- To control the flow of inference values, use the blocks on the Logic Gates palette and the Temporal Gates palette in the Inference Blocks menu.
- To control the flow of control signals, use the blocks on the Control Actions palette in the Action Blocks menu.

Data Delay



The Data Delay block delays passing its input value for a specified amount of time. It does not modify its input value. Specify the length of the delay with the Delay attribute.

Handling Multiple Signals

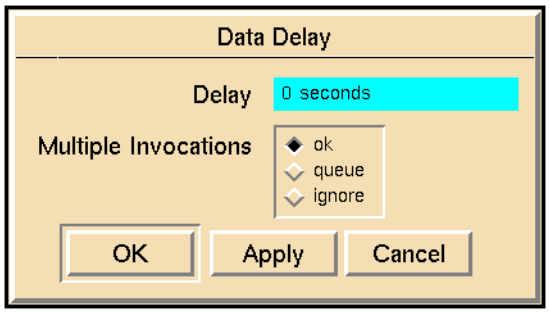
If Multiple Invocations is ok, the block can begin a delay while in the midst of delaying another value. If Multiple Invocations is queue, the second and subsequent delays will be started after the previous delay is finished, using the current input value. If Multiple Invocations is ignore, the block handles only one delay at a time, and inputs received during another delay are ignored. We recommend specifying Multiple Invocations for a Data Delay as either ok or ignore.

Resetting

When you reset a Data Delay block, values being delayed are lost.

Configuring

This is the configuration panel for the Data Delay block.



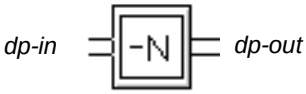
Attribute	Description
Delay	The amount of time to delay passing the value.
Multiple Invocations	See “Specifying How to Handle Multiple Values” on page 92 in the *GDA User’s Guide*.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
The Inference Delay block	page 363	*Reference Manual*
The Control Delay block	page 441	*Reference Manual*

Data Shift



The Data Shift block delays its input by a certain number of inputs; it waits to pass an input value until it has received more data. You specify how many more input values it must receive with the Sample Size attribute.

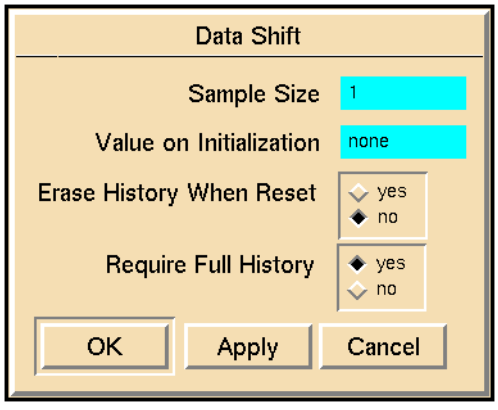
Specifying How to Delay Values

The block buffers its inputs in a first-in first-out queue of length Sample Size. When it receives a value, it deletes the value at the beginning of the queue and passes that value.

If the block does not have a value that satisfies its conditions, it passes no value. For example, when the block does not have Sample Size points yet, it outputs nothing.

Configuring

This is the configuration panel for the Data Shift block.



Attribute	Description
Sample Size	The number of points the block must receive before passing a value.

Attribute	Description
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.
Erase History When Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.

Example

This table shows sample input and output for a Data Shift block with a Sample Size of 3. The first row is the values it received. The second row is the values it passed on.

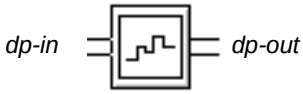
Input	1.3	2.3	4.5	5.0	7.2
Output				1.3	2.3

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Data Delay	page 130	*Reference Manual*
Sample and Hold	page 134	*Reference Manual*

Sample and Hold

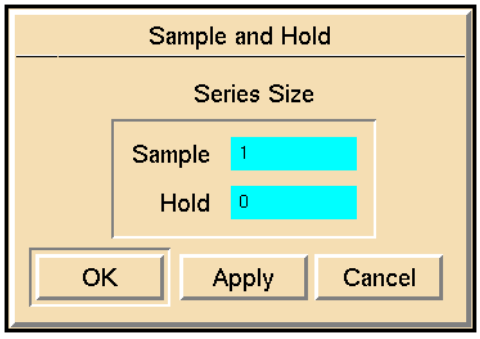


The Sample and Hold block reduces the amount of data that flows in a path by selectively passing and discarding values. It is also useful for removing auto-correlation in a data series.

Specify the number of values to pass with the attribute Sample Series Size. Specify the number of values to discard with the attribute Hold Series Size. The block passes Sample Series Size values, discards Hold Series Size values, and then starts over.

Configuring

This is the configuration panel for the Sample and Hold block.



Attribute	Description
Sample Series Size	The number of values to pass before discarding one or more values.
Hold Series Size	The number of values to discard before passing more values.

Example

If Sample Series Size is 2 and Hold Series Size is 3, the block passes 2 out of every 5 values. In the sample below, the block passes only the marked values:

```
3  3        3  3        3  3        3  3
5  7  9  2  3  6  8  5  4  9  0  5  2  1  5  8  7  8  8  9
```

To pass one out of 10 input values, set Sample Series Size to 1 and Hold Series Size to 9. In the sample below, the block passes only the marked values:

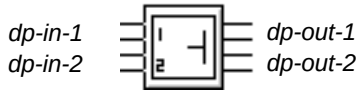
```
3                       3
5  7  9  2  3  6  8  5  4  9  0  5  2  1  5  8  7  8  8  9
```

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Filters Palette	page 63	*Reference Manual*

Data Synchronize



The Data Synchronize block synchronizes two data paths by passing their values only after it receives values on each data path. After it passes two values, it will pass values again only after it receives a value on each data path.

This block is useful when you must make sure that all the data values in a flow diagram are current. For example, if two Entry Points send out values at slightly different times, blocks connected to both of them will evaluate twice, once for each value. When the blocks evaluate, they will be using a current value and an obsolete value, so the results may be inaccurate. If the Entry Points are connected to a Data Synchronize block, however, the values will go out simultaneously and the blocks connected to them will evaluate only once with two current values.

Resetting

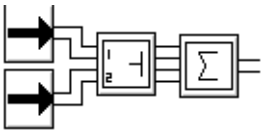
When you reset a Data Synchronize block, it passes any pending input value.

Configuring

The Data Synchronize block has no configurable attributes.

Example

This figure shows a flow diagram that uses a Data Synchronize block:

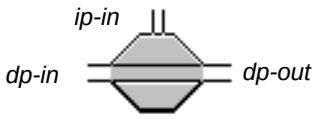


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The True if Expired block	page 297	*Reference Manual*
The Event Sequence Gate block	page 350	*Reference Manual*
The Control Synchronize block	page 461	*Reference Manual*

Data Inhibit



The Data Inhibit block lets an inference path turn a data path on and off.

When the status value of the block's input inference path matches the value of the attribute Trigger On, the block inhibits the input data value. If Value on Initialization has a value, it passes that value. Otherwise it passes nothing.

When the status value of the inference path no longer matches Trigger On, the block passes the current value of the input data path and continues to pass input data values normally.

If the block's inference path has not received a value yet (that is, it has a quality of no-value), the block passes nothing even when it receives a value from its data path.

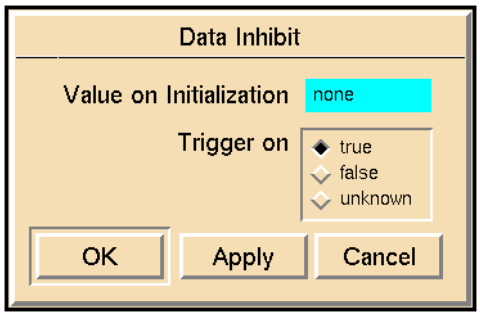
GDA evaluates this block whenever the data path receives a new value or when the data inference path changes to or from the Trigger On value. It also evaluates the block when the data inference path changes from one non-trigger value to another, for example, from false to unknown.

Resetting

When you reset a Data Inhibit Block, the block does not pass a value until it receives a value from its inference input path, even if it receives a value from its input data path.

Configuring

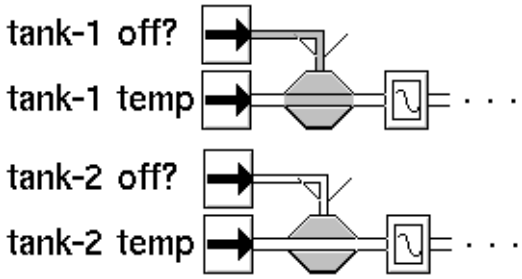
This is the configuration panel for the Data Inhibit block.



Attribute	Description
Value on Initialization	See “Specifying an Initial Data Value” on page 83 in the *GDA User’s Guide*.
Trigger On	The truth value that causes the block to inhibit data.

Example

This figure shows a portion of a flow diagram that uses two Data Inhibit blocks to test whether a tank is on before analyzing its temperature. Tank-1 is off and the path that crosses through the middle of the Data Inhibit block is filled in to show that it is inhibiting its input. Tank-2 is on and the middle of the Data Inhibit block is empty to show that the block is passing along its input.

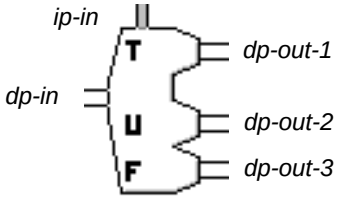


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
The Inference Inhibit block	page 303	*Reference Manual*
The Control Inhibit block	page 447	*Reference Manual*

Data Switch



The Data Switch lets an inference value choose which path to send a data value down. It is useful when you need to perform different diagnostics depending on some condition.

The value of the top input path (ip-in) determines where the value of left input path (dp-in) will go. The following table shows which output path the block chooses:

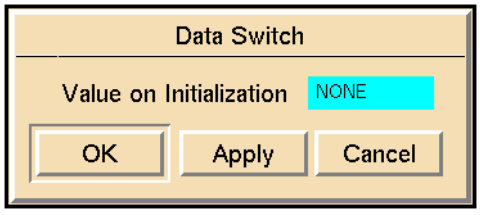
If the top input is…	The left input is passed on the port labeled…
.true	“T” (dp-out-1)
unknown	“U” (dp-out-2)
.false	“F” (dp-out-3)

The attribute Value on Initialization determines the value of the two output paths that do not get the input value. Those paths have the value of Value on Initialization, with new timestamps.

The Data Switch block only passes values when the data value changes, not when the inference value changes.

Configuring

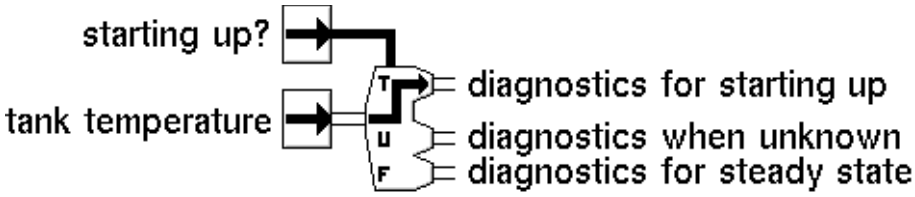
This is the configuration panel for the Data Switch.



Attribute	Description
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

This figure shows a diagram that determines which tank diagnostics to run, depending on whether the process is starting up or not. Notice that an arrow on the Data Switch icon shows you which path the block is choosing.



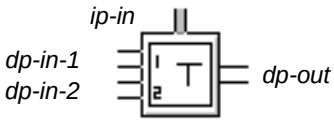
Since the top input path is .true, the output path labeled "T" has the same value as the left input path. The other output paths are none, the value of Value on Initialization.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Inference Switch block	page 305	*Reference Manual*
The Control Switch block	page 454	*Reference Manual*

Data Select



The Data Select block selects one of two data inputs, depending on the value of its inference input. This table describes how the block chooses which value to pass:

If the input status value is…	Then the block passes…
.true	The top input data value (dp-in-1)
.false	The bottom input data value (dp-in-2)
unknown	Nothing

Note that the block passes a value only if its input status value is .true or .false. If the input status value is unknown, the Data Select block passes nothing and does not evaluate any attached chart or graph capabilities.

GDA evaluates the block whenever any input (a data input or the inference input) receives a new value. The block passes a new value, however, only if the value on the selected data input path is different from the output path value or has the same value but a newer timestamp than the value on the output path.

Configuring

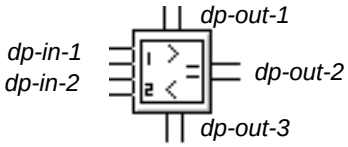
The Data Select block has no configurable attributes.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Conditional Test



The Conditional Test block passes the top input value to one of three different output paths, depending on how the top and bottom input values compare. This table shows how the block chooses the path along which to pass its value.

If…	Then the top input value is passed along the…
top input > bottom input	Top output path (dp-out-1)
top input = bottom input	Middle output path (dp-out-2)
top input < bottom input	Bottom output path (dp-out-3)

Using Fuzzy Logic

The Conditional Test block has an attribute called Equivalence Band, which allows you to use fuzzy logic to determine equivalency. The default value is none, which means that the block does not use fuzzy logic. If this attribute is a non-zero floating-point number, the value of the top input path is passed along the middle output path (dp-out-2) when the following condition is met, where EB is the Equivalency Band:

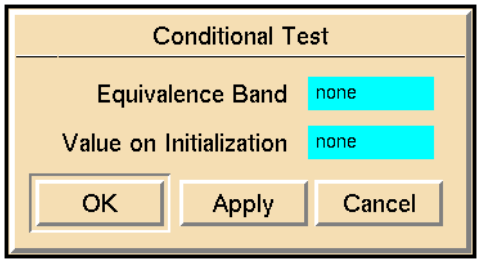
$$(\text{dp-in-2} - \text{EB}/2) \leq \text{dp-in-1} \leq (\text{dp-in-2} + \text{EB}/2)$$

Resetting

When you reset a Conditional Test block, it passes the attribute Value on Initialization to all its output paths.

Configuring

This is the configuration panel for the Conditional Test block.



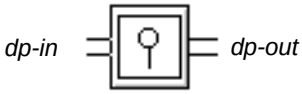
Attribute	Description
Equivalence Band	The amount of uncertainty that the block applies to the top input value to determine equivalency. The number represents a positive and negative uncertainty band around the input value.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Inference Switch block	page 305	*Reference Manual*

Data Output



The Data Output block is obsolete, because you can connect variables and parameters directly to the output of any block. For information on how to do this, see "Using Variables and Parameters" on page 105 in the *GDA User's Guide*.

Use this block only when you cannot attach a data path directly to the target variable or parameter. For example, if the variable or parameter is an attribute of an object or if you want variables and parameters to appear on a separate workspace, use the Data Output block.

When you set the Data-server attribute in the table of the variable or parameter to inference engine, GDA uses the G2 conclude action to change values locally within GDA. When you set the Data-server attribute to anything other than inference engine, however, GDA uses the G2 set action to change values in the external system outside of G2.

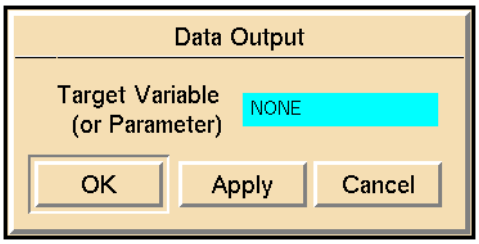
Note You can set Target Variable to an expression that evaluates to a G2 variable or parameter. For more information, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.

This block is especially useful for setting external setpoints or controlling external events.

The "Entry Points" on page 3 perform the opposite action of the Data Output block: they pass information from G2 parameters and variables into GDA diagrams.

Configuring

This is the configuration panel for the Data Output block.



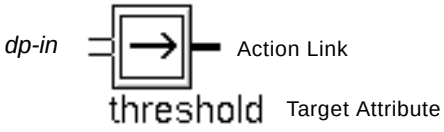
Attribute	Description
Target Variable	The name of the variable or parameter whose value the block updates.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Using Variables and Parameters	page 105	*User's Guide*
Entry Points	page 7	*Reference Manual*
Network Entry Points	page 589	*Reference Manual*

Set Attribute



The Set Attribute block sets the value of an attribute to the block's input data value. Specify the object whose attribute you would like to set by connecting the Set Attribute block's action link to it. You can connect the action link to any G2 object or block. Specify the attribute you would like to set by setting Target Attribute to its name.

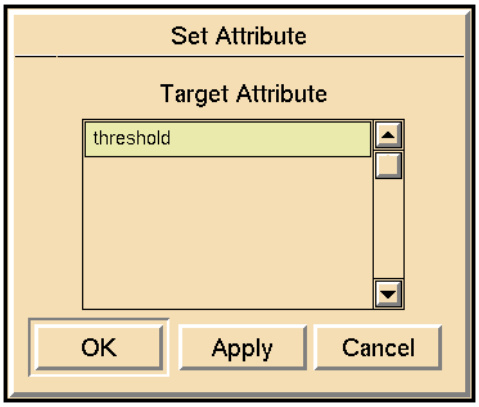
When you connect a block to the Set Attribute block's action link, the configuration panel displays a list of all attributes defined by the connected block. Thus, you should always connect the Action Link to the target *before* configuring the block.

Take the following precautions when using the Set Attribute block:

Caution When you use the Set Attribute block, be sure that the changes to the attribute value leave the target block in a valid state; otherwise, errors will result. For example, if you use the Set Attribute block to set the Upper Threshold of an observation block to a number that is less than the Lower Threshold, you will receive a warning and a runtime error might result.

Configuring

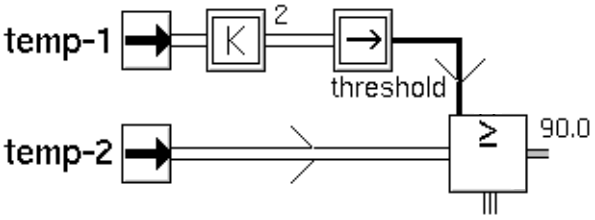
This is the configuration panel for the Set Attribute block.



Attribute	Description
Target Attribute	An attribute of the block connected to the action link, whose value the Set Attribute block sets.

Example

The following figure uses the Set Attribute block to test whether the temperature of tank-2 is more than twice the temperature of tank-1:

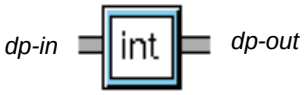


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
The "Block Actions" Palette	page 419	*Reference Manual*

Integer



The Integer block coerces data from a type quantity to a type integer. The block is intended to be used primarily to precede the Set Attribute block to prevent errors where the attribute in question is an integer.

Configuring

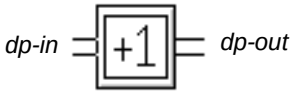
The Integer block has no configurable attributes.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Set Attribute block	page 147	*Reference Manual*

Input Counter



The Input Counter block counts the number of inputs it receives and passes the number.

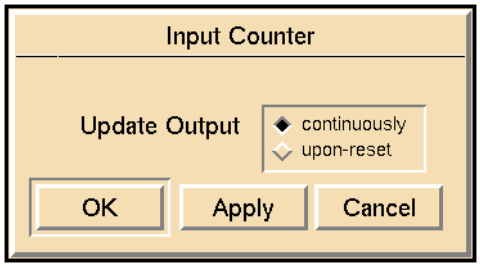
The attribute Update Output controls when the block passes a value and what happens when you reset the block. This table lists the attribute's two possible values:

If Update Output is…	The block passes a value when…	And does this when reset...
continuously	It receives a value	Resets the count and passes 0.
upon reset	It is reset	Passes the count and resets it.

Note You can reset a block with the Reset block on page 428.

Configuring

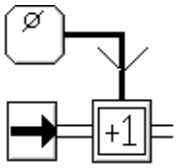
This is the configuration panel for the Input Counter block.



Attribute	Description
Update Output	Whether the block passes a value when it receives it (continuously) or only when the block is reset (upon reset).

Example

This figure shows an Input Counter block attached to an Entry Point and a Reset Block action block:



Suppose the following happens:

1. The Entry Point passes 5 values.
2. The Reset Block resets the Input Counter.
3. The Entry Point passes 3 values.
4. The Reset Block resets the Input Counter.

This table lists what the Input Counter block passes depending on the value of Update Output:

If Update Output is…	Before 1st reset, it passes…	Before 2nd reset, it passes…	After 2nd reset, it passes...
continuously	1, 2, 3, 4, 5	0, 1, 2, 3	0
upon reset		5	3

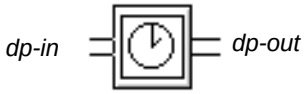
Notice that when Update Output is upon-reset, the block does not pass anything before it is reset.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Inference Counter block	page 372	*Reference Manual*
The Control Counter block	page 449	*Reference Manual*

Data Time Stamp



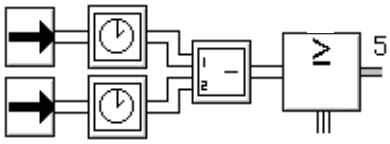
The Data Time Stamp block passes the Collection-time of its input value. This block is useful when you need to know how old a value is or when an event occurred. Collection-time is an attribute of a data path.

Configuring

The Data Time Stamp block has no configurable attributes.

Example

This figure tests whether the top entry point's data is more than 5 seconds newer than the bottom entry point's data:



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Event Window Gate block	page 353	*Reference Manual*

7

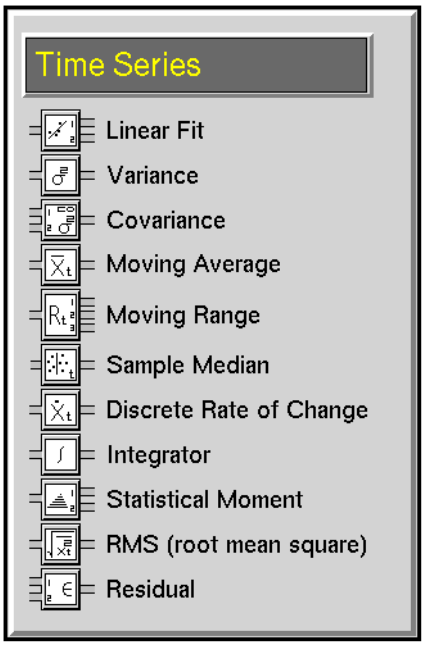
Time Series

Describes the blocks that perform operations on a history of values.

Introduction

GDA provides you with several blocks that let you perform operations on a history of values. You can find the average value and other statistical properties of the history, such as the median, variance, skew, and kurtosis. You can integrate over the history, find the line that best fits the history, and more.

You can find the Time Series palette under the Data submenu of the Palettes menu:



Averaging Over a History

These blocks compute the average and median value of a history of values:

- The Moving Average block on page 165 computes the average of the history of values.
- The Sample Median block on page 170 computes the median value in the history of values.

Computing Statistical Properties

These blocks compute other statistical properties:

- The Variance block on page 159 computes the variance of the history of values.
- The Statistical Moment block on page 178 computes the skew and kurtosis (also called the third and fourth statistical moments) of the history of values.
- The Moving Range block on page 167 computes the maximum, minimum, and range (the difference between the maximum and minimum) of the history of values.
- The RMS (root mean square) block on page 181 computes the root mean square of the history of values.

Comparing Two Values

These blocks keep a history for two input values and compare them:

- The Covariance block on page 161 computes either the covariance or the correlation coefficient for the values. The covariance tells you how closely the two values are related. The correlation coefficient scales the covariance to a number between -1.0 and 1.0.
- The Residual block on page 183 computes the sum of errors over a history of pairs of values. You can choose among three different ways of computing the error.

Performing Linear and Polynomial Functions

These blocks perform linear and polynomial operations on their input values:

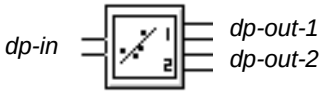
- The Linear Fit block on page 156 computes the current best fit for a line over the history of values. It passes on the slope and best fit for that line.
- The Discrete Rate of Change block on page 172 computes the instantaneous rate of change (or slope) for its input value.
- The Integrator block on page 175 computes the area under the curve of the history of its input values.

See Also

Here are some other blocks that perform similar calculations:

- For gates that operate on a history of belief values, use the blocks on the Temporal Gates palette in the Inference Blocks menu.
- For gates that operate on any number of input data values, use the blocks on the Arithmetic palette and Functions palette in the Data Blocks menu.

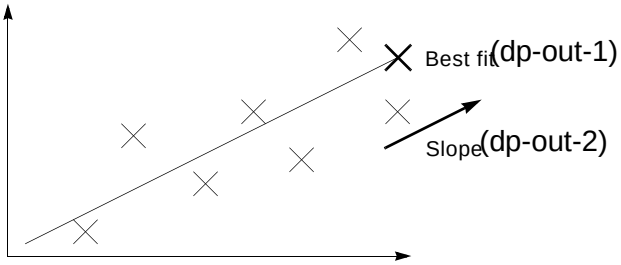
Linear Fit



The Linear Fit block calculates the current best fit for a line over the history of values. It passes the best-fit value for the current input and the rate of change of the history (the slope), as shown in this table:

The block passes the…	On this port…
Best fit	Top port (dp-out-1)
Slope	Bottom port (dp-out-2)

This figure shows how the Linear Fit block might fit a line to a history of values. The light crosses are the values in the history. The dark cross is the best fit for the latest value.



This block is frequently used with a Linear Prediction block, which predicts a future value for a point on a line. The Linear Prediction block's input values are the same as the Linear Fit block's output values (a point on a line and the line's slope) so you can connect a Linear Fit block directly to a Linear Prediction block.

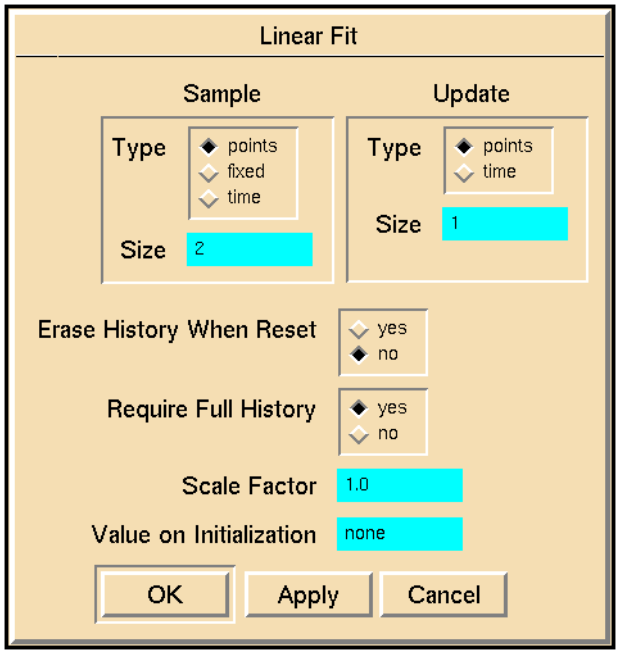
Specifying How to Convert the Slope

To multiply the slope of the best-fit line by a constant value, set the attribute Scale Factor to that value. The block multiplies the slope before passing it.

The Scale Factor is especially useful when you need to convert the slope from one set of units to another. For example, if Scale Factor is 60, the block would convert change per second to change per minute.

Configuring

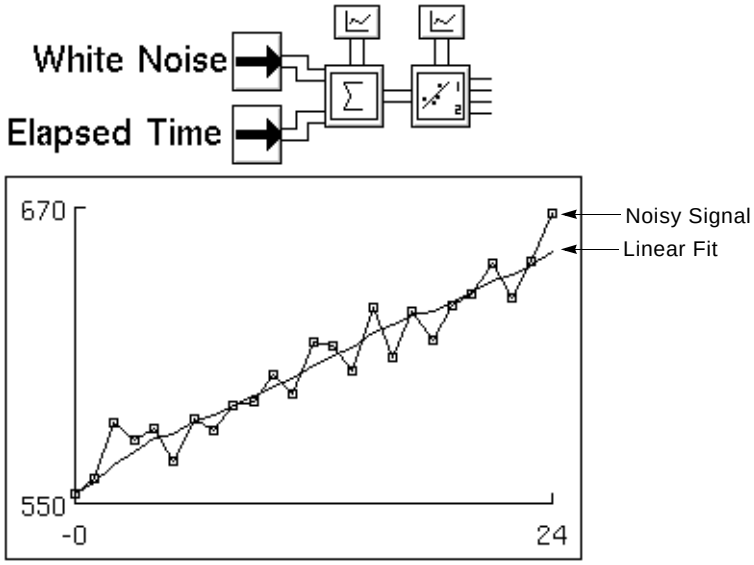
This is the configuration panel for the Linear Fit block.



Attribute	Description
Sample Type and Sample Size	See “Specifying the Size of the History” on page 85 in the *GDA User’s Guide*.
Update Type and Update Size	See “Specifying When to Propagate Data” on page 88 in the *GDA User’s Guide*.
Erase History when Reset	See “Specifying What Happens to History Upon Reset” on page 90 in the *GDA User’s Guide*.
Require Full History	See “Specifying What to Do With Partial History” on page 91 in the *GDA User’s Guide*.
Scale Factor	A constant value by which the block multiplies the slope of the best-fit line before passing it on the bottom output port (dp-out-2).
Value on Initialization	See “Specifying an Initial Data Value” on page 83 in the *GDA User’s Guide*.

Example

The following figure shows a Linear Fit block connected to blocks that produce a noisy signal. The Linear Fit block has a history of 25 points. The noisy signal is the sum of an Elapsed Time block and a White Noise block. In the chart, the line marked “Linear Fit” contains the best-fit points that the Linear Fit block passed.

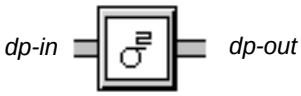


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Maintaining a History of Values	page 84	*User’s Guide*

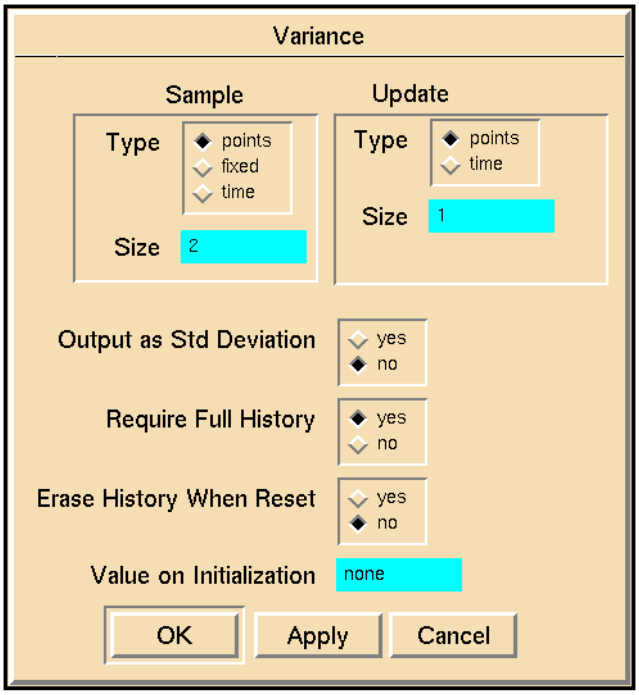
Variance



The Variance block passes on either the statistical variance or the standard deviation of the history of its input value. You can choose which it passes with the attribute Output as Std Deviation. If Output as Std Deviation is yes, the block passes the standard deviation. Otherwise, it passes the variance, which is the square of the standard deviation.

Configuring

This is the configuration panel for the Variance block.



Attribute	Description
Sample Type and Sample Size	See “Specifying the Size of the History” on page 85 in the *GDA User’s Guide*.
Update Type and Update Size	See “Specifying When to Propagate Data” on page 88 in the *GDA User’s Guide*.

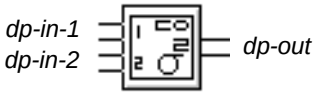
Attribute	Description
Output as Std Deviation	Whether the block outputs the value as a standard deviation (yes) or variance (no).
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

Covariance



The Covariance block passes on either the covariance or the correlation coefficient for a history of pairs of values. The correlation coefficient is the covariance scaled to a value between -1.0 and 1.0. You can choose which it passes with the attribute Covariance Scaling. If Covariance Scaling is yes, the block passes the correlation coefficient. Otherwise, it passes the covariance.

The covariance is especially useful for keeping track of the error between a pair of redundant sensors. The correlation coefficient is especially useful for sensors which measure different, but related things.

Using the Correlation Coefficient

The correlation coefficient lets you use test whether two measurements are correlated. You can use this information to determine whether your input data is valid. For example, suppose that when a tank's temperature goes up, the tank's pressure also goes up. You can use this correlation to determine if the tank's sensors are working. If the temperature and pressure sensors have a high correlation coefficient, you know that the sensors are working. If the sensors have a low correlation coefficient, you know that at least one is broken.

This coefficient ranges from -1.0 to 1.0, as described in this table:

A coefficient of…	Means the input values are…
-1.0	Negatively correlated. (If one input is high, the other is low.)
0.0	Not correlated.
1.0	Positively correlated. (If one input is high, the other is high. If one input is low, the other is low.)

Specifying Whether Values Are Concurrent

Usually, you want to compare the input values from the two different ports only if those values were created at approximately the same time. However, values that were created at the same time may arrive at slightly different times due to computer speed or network speed. This block considers two values to be

concurrent if they arrive within the period of time you specify in the attribute Concurrency Window.

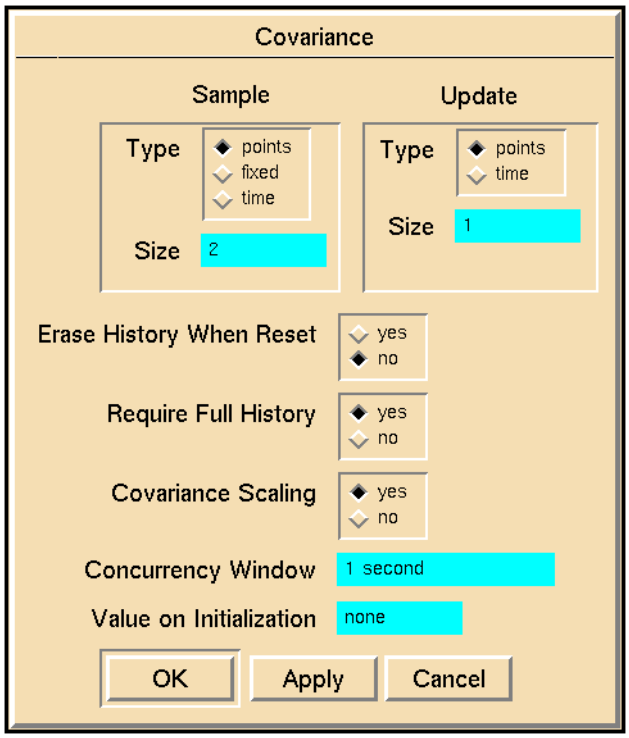
For example, suppose that Concurrency Window is 1 second. If you receive a value in the top port and then a half second later receive a value in the bottom port, the block considers the two values to be concurrent, stores them in the block's history as a pair, and passes a new value. However, if you receive a value in the bottom port, and one second passes with no value received on the top port, the block discards the value it received on the bottom port, and does not pass on a new value.

If you receive several values over one port before you receive a value over the other port, the block discards all but the latest value it received.

Note If you are testing a diagram by entering data manually (for example, by using the override menu choice), make sure the Concurrency Window is large enough so you can enter values at a reasonable pace.

Configuring

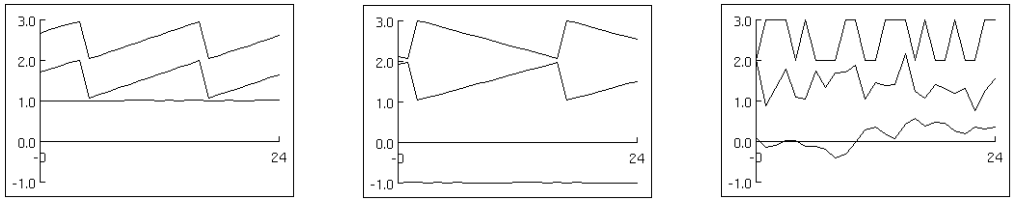
This is the configuration panel for the Covariance block.



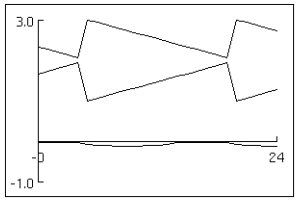
Attribute	Description
Sample Type and Sample Size	See “Specifying the Size of the History” on page 85 in the *GDA User’s Guide*.
Update Type and Update Size	See “Specifying When to Propagate Data” on page 88 in the *GDA User’s Guide*.
Erase History when Reset	See “Specifying What Happens to History Upon Reset” on page 90 in the *GDA User’s Guide*.
Require Full History	See “Specifying What to Do With Partial History” on page 91 in the *GDA User’s Guide*.
Covariance Scaling	Whether the block passes the correlation coefficient (yes) or the covariance (no).
Concurrency Window	The amount of time that the block applies to the top input value to be considered concurrent with the bottom input value.
Value on Initialization	See “Specifying an Initial Data Value” on page 83 in the *GDA User’s Guide*.

Example

The following figure shows three charts of correlation coefficients. In each graph, the correlation coefficient is plotted at the bottom of the graph, and the two input signals are plotted at the top. The Covariance block’s history contains 10 points. The first graph shows two signals with a positive correlation. The second shows two signals with a negative correlation. The third shows two signals with almost no correlation.



This figure shows a graph of a covariance using the same signals as in the second graph above. The Covariance block's history contains 10 points.

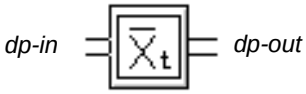


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

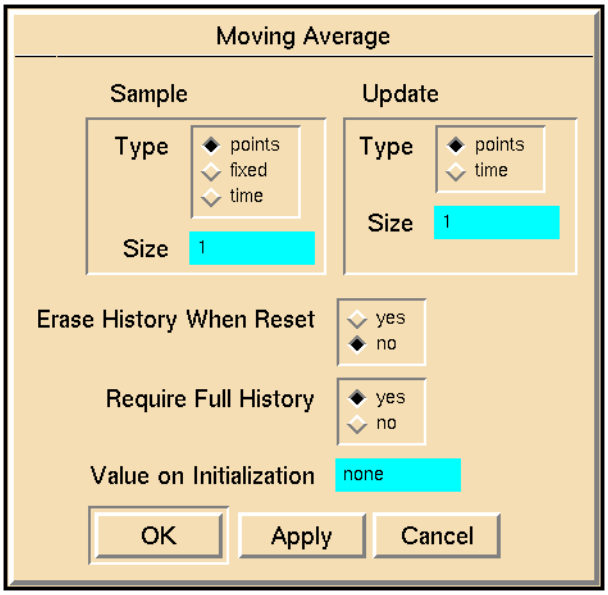
Moving Average



The Moving Average block passes the average of the history of its input values.

Configuring

This is the configuration panel for the Moving Average block.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.

Attribute	Description
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

If the block's history contains 3, 0, 5, 8, 2, and 6, it passes on 4 or:

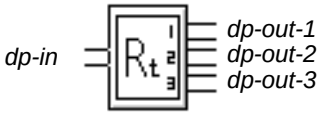
$$\frac{3 + 0 + 5 + 8 + 2 + 6}{6}$$

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
Average Input Value	page 104	*Reference Manual*
Average Belief Gate	page 344	*Reference Manual*

Moving Range

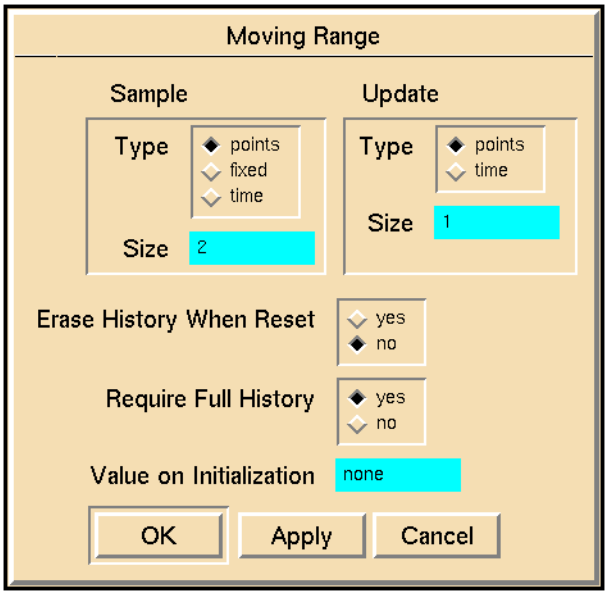


The Moving Range block passes on the maximum, minimum, and range (the difference between the maximum and the minimum) of the block's history of values, as shown in the following table:

The block passes this value…	On the…
Maximum	Top port (dp-out-1)
Range (Maximum – Minimum)	Middle port (dp-out-2)
Minimum	Bottom port (dp-out-3)

Configuring

This is the configuration panel for the Moving Range block.



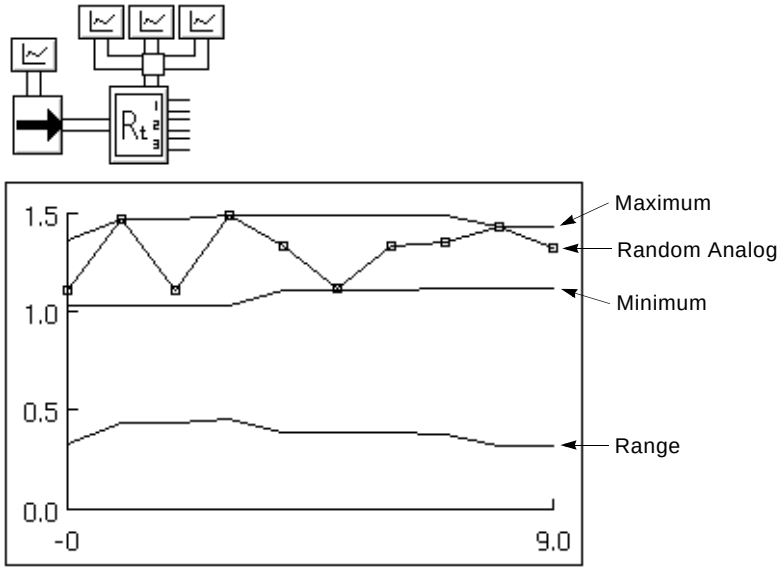
Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

If a Moving Range block's history contains 1.25, 1.39, 1.02, 1.14, 1.45, and 1.27, it passes the values in this table:

The block passes this value…	On the…
1.45 (Maximum)	Top port (dp-out-1)
0.43 (Range)	Middle port (dp-out-2)
1.02 (Minimum)	Bottom port (dp-out-3)

The following figure charts the output of a Moving Range block attached to a Random Analog block. The Random Analog block is generating random values between 1.0 and 1.5. The Moving Range block is keeping a history of 5 points and passing a value. whenever it receives a new point. The graph marks the output of the Random Analog block with rectangles.

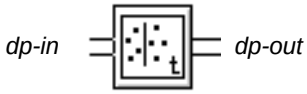


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
Low Selecting, High Selecting	page 108	*Reference Manual*
Low Limiting, High Limiting	page 110	*Reference Manual*
The Max Belief Gate	page 346	*Reference Manual*
The Min Belief Gate	page 348	*Reference Manual*

Sample Median

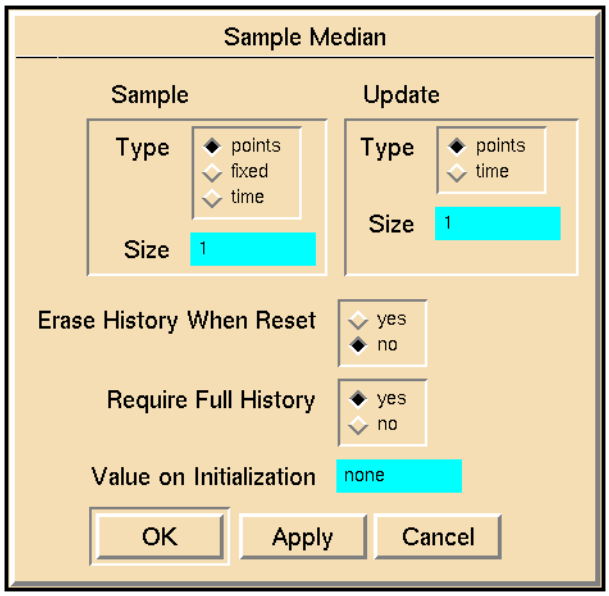


The Sample Median block passes the median of the history of its input values.

If the Sample Median block has an odd number of history values, it passes the middle value. If the block has an even number of history values, it passes the lesser of the two middle values.

Configuring

This is the configuration panel for the Sample Median block.



Attribute	Description
Sample Type and Sample Size	See “Specifying the Size of the History” on page 85 in the *GDA User’s Guide*.
Update Type and Update Size	See “Specifying When to Propagate Data” on page 88 in the *GDA User’s Guide*.
Erase History when Reset	See “Specifying What Happens to History Upon Reset” on page 90 in the *GDA User’s Guide*.

Attribute	Description
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

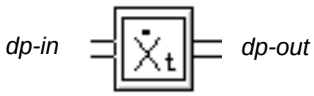
If the history of a Sample Median block has the five points 0.7, 0.9, 0.3, 0.7, and 0.5, it passes on the value 0.7, the middle value. If the history of a Sample Median block has the four points 0.3, 0.9, 0.6, and 0.8, it passes on the value 0.6, the lesser of 0.6 and 0.8.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
The Median Input Value block	page 106	*Reference Manual*

Discrete Rate of Change



The Discrete Rate of Change block computes the instantaneous rate of change (or slope) for its input value using a general least squares fit method. You can specify whether the polynomial order it uses is quadratic or cubic and how many points it calculates the rate of change over.

Specifying the Number of Points

The Discrete Rate of Change block requires a full history of 5 or 7 points. In the configuration panel, be sure to click on Yes to require full history.

Note Unlike other blocks on the Time Series palette, the Discrete Rate of Change block does not let you choose how it keeps its history. It does not have the attributes Sample Type or Sample Size, and the attribute Number of Points must be either 5 or 7.

Specifying the Polynomial Order

To specify the polynomial order that the block uses, set the attribute Polynomial Order to either 2 or 3, as this table shows:

To use this order…	Set Polynomial Order to…
Quadratic	2
Cubic	3

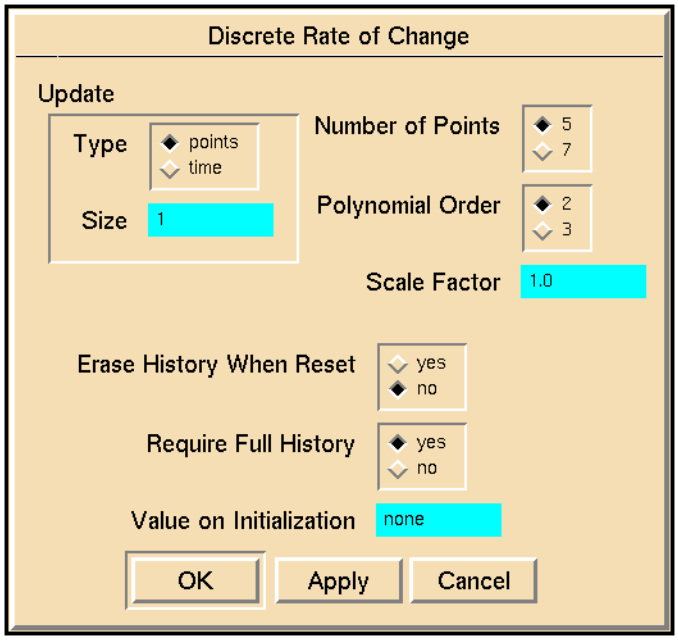
Specifying How to Convert the Slope

To multiply the slope by a constant value, set the attribute Scale Factor to that value. The block multiplies the slope before passing a value.

The Scale Factor is especially useful when you need to convert the slope from one set of units to another. For example, if Scale Factor is 60, the block would convert change per second to change per minute.

Configuring

This is the configuration panel for the Discrete Rate of Change block.



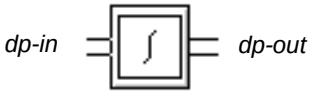
Attribute	Description
Update Type and Update Size	See “Specifying When to Propagate Data” on page 88 in the *GDA User’s Guide*.
Number of Points	The number of points that the block uses to perform its filtering.
Polynomial Order	Whether the block uses a cubic or quadratic polynomial to compute the rate of change.
Scale Factor	A constant value by which the block multiplies the slope before passing the output value.
Erase History when Reset	See “Specifying What Happens to History Upon Reset” on page 90 in the *GDA User’s Guide*.
Require Full History	See “Specifying What to Do With Partial History” on page 91 in the *GDA User’s Guide*.
Value on Initialization	See “Specifying an Initial Data Value” on page 83 in the *GDA User’s Guide*.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
Quadratic Filter, Cubic Filter	page 81	*Reference Manual*
The Linear Fit block	page 156	*Reference Manual*

Integrator



The Integrator block passes on the Euler integral of the block's history of values. If you were to graph the history of values, with the X axis plotted as the time the values were received and the Y axis as the actual values, the integral is the area beneath this curve.

The block uses a trapezoidal integration algorithm. In addition, when Sample Type is time, the block interpolates to include the area between the oldest valid data point and the sample limit. See the example.

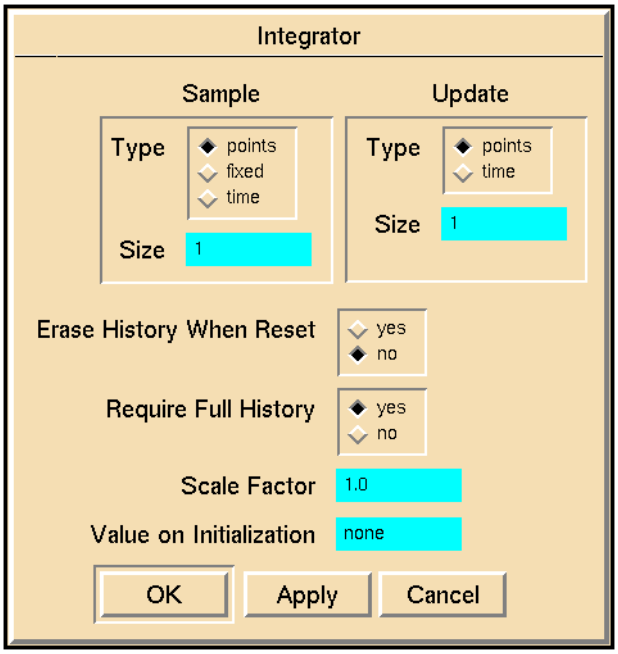
Specifying How to Convert the Area

To divide the area by a constant value, set the attribute Scale Factor to that value. The block divides the area before passing it.

The Scale Factor is especially useful when you need to convert the area from one set of units to another. For example, if Scale Factor is 60, the block converts the time over which the area is calculated from minutes to seconds, assuming the Sample Type is time.

Configuring

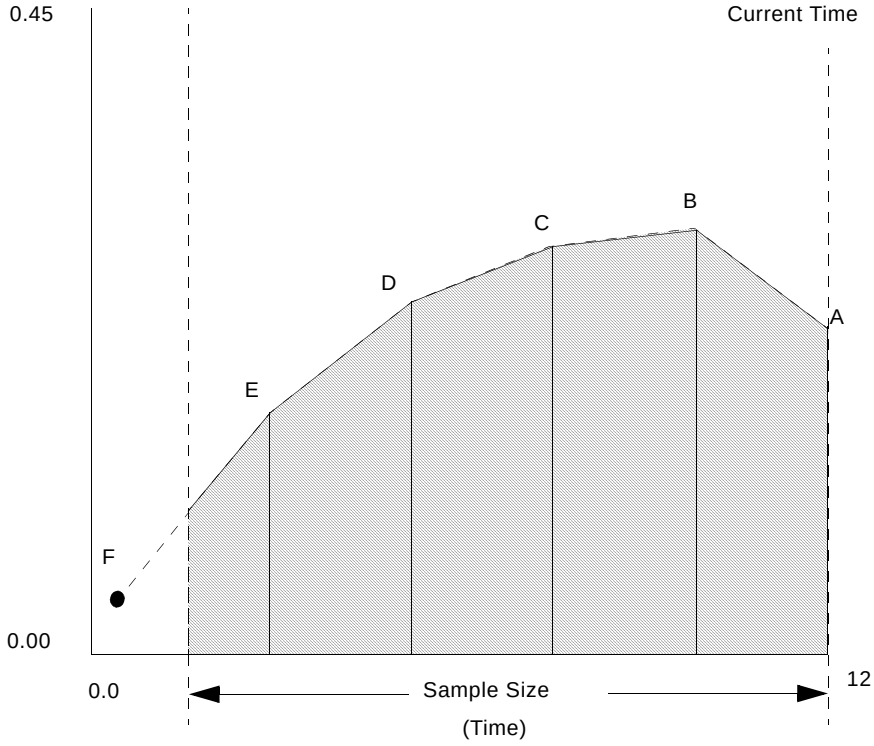
This is the configuration panel for the Integrator block.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Scale Factor	A constant value by which the block divides the area before it passes the output value.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

The following figure illustrates the new trapezoidal integration algorithm. Notice that one point (F) is beyond the valid range of the sample size. It is retained and GDA performs a linear interpolation to determine an area for the time between (E) and (F).



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
The Median Input Value block	page 106	*Reference Manual*

Statistical Moment

The Statistical Moment block helps you determine the amount by which the block's history of values deviates from a normal distribution. It is especially useful when you use it with a block that expects its input values to be normally distributed, such as most SPC blocks.

The Statistical Moment block passes the third and fourth moments as shown in this table:

The block passes the…	On the…
Skew (third moment)	Top port (dp-out-1)
Kurtosis (fourth moment)	Bottom port (dp-out-2)

If the block's input is normally distributed, the skew is between $-7.35 \times \text{Sample-size}^{-0.5}$ and $7.35 \times \text{Sample-size}^{-0.5}$, and the kurtosis is between $3 - (14.7 \times \text{Sample-size}^{-0.5})$ and $3 + (14.7 \times \text{Sample-size}^{-0.5})$.

Use these equations to calculate the appropriate limits for Observation blocks.

Note Make sure the block's history contains at least 5 points. You will usually want at least 9 points.The Statistical Moment block needs a large history to produce accurate values.

The Statistical Moment block uses these equations to compute its output value. The variables in these equations are explained in the table that follows.

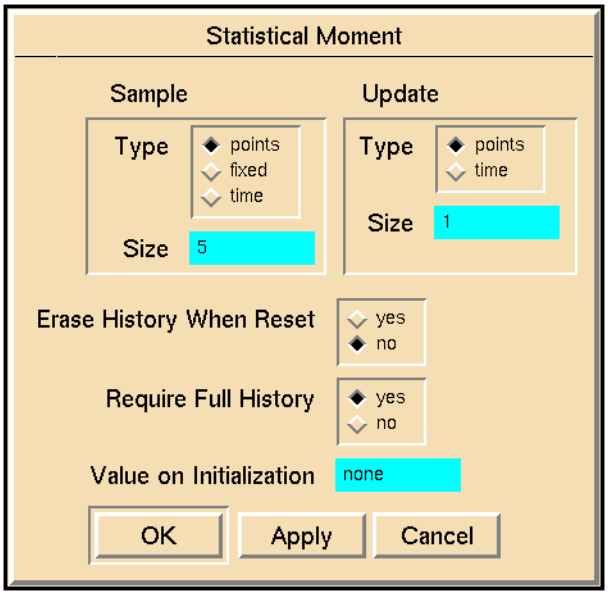
$$\text{skew} = \frac{1}{\text{sample} - \text{size}} \sum_{i=1}^{\text{sample} - \text{size}} \left[\frac{x_i - \bar{x}}{S}\right]^3$$

$$\text{kurtosis} = \left\{\frac{1}{\text{sample} - \text{size}} \sum_{i=1}^{\text{sample} - \text{size}} \left[\frac{x_i - \bar{x}}{S}\right]^4\right\} - 3$$

This variable…	Is…
S	The standard deviation of the block's history.
x	An entry in the block's history.
$\bar{x}$	The average of the block's history

Configuring

This is the configuration panel for the Statistical Moment block.

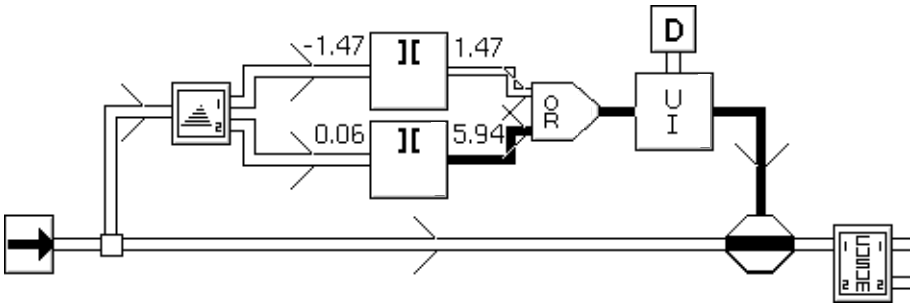


Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.

Attribute	Description
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

The Statistical Moment block in this figure helps you determine whether the Entry Point is passing normally distributed data. The Out of Range Observations test whether the skew and kurtosis are within the proper range for a sample size of 25 points. If either the skew or kurtosis is outside its range, the Inference Query block asks the operator whether to inhibit data from passing.

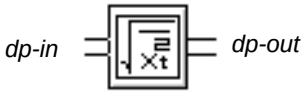


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

RMS (root mean square)



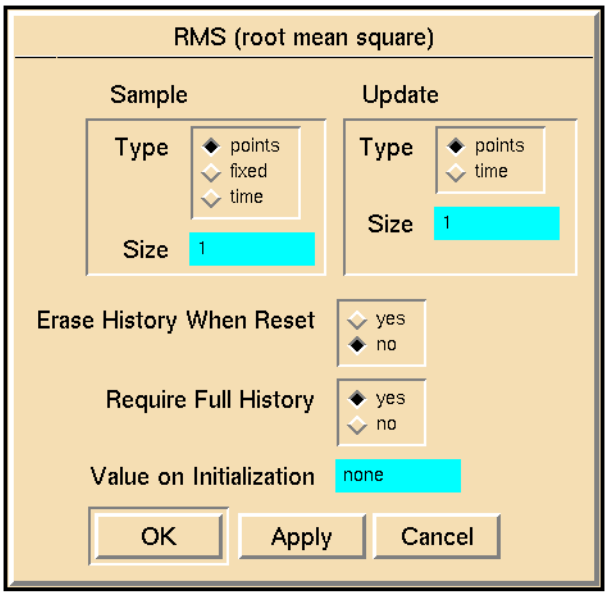
The RMS block calculates the root mean square of the block's history of values. It uses this equation to compute its output value:

$$\text{output}_n = \sqrt{\frac{\sum_{n=1}^{\textit{Sample-size}} \text{input}_n^2}{\textit{Sample-size}}}$$

This block is especially useful for computing the average error value when the error can have both positive and negative values.

Configuring

This is the configuration panel for the RMS block.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

This table shows possible values for a RMS block. The input values are the same as the differences of the data pairs used in the example of the Residual block on page 183.

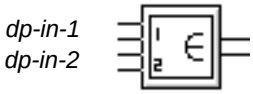
If the block receives these values…	It passes this value…
1, 6, 3, 4, 2	3.633
1, -6, 3, -4, 2	3.633
1, -0.6, 3, -0.4, 2	1.704

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

Residual



The Residual block computes the sum of errors over a history of pairs of input values. You can choose among three different ways of computing the error. This block is especially useful for keeping track of the error between a measured value and the expected value.

Specifying How to Compute the Error

The attribute Error Type controls how the block computes the error for a pair of values. Error Type can be any of the symbols in this table:

If Error Type is…	The block computes the error with this equation…
sum	$input_{top} - input_{bottom}$
abs (absolute value)	$\lvert input_{top} - input_{bottom} \rvert$
sse (sum of squared errors)	$(input_{top} - input_{bottom})^2$

Here are some facts to keep in mind when you set the Error Type:

- When Error Type is abs the block counts all errors equally, regardless of the sign of the errors.
- When Error Type is sum, successive positive and negative errors cancel each other out. For example, the residual for the differences -1.0 and +1.0 is the same as the residual for the differences -100.0 and +100.0.
- When Error Type is sse, the largest input errors tend to dominate the block's output value.

Specifying Whether Values are Concurrent

Usually, you want to compare values from the two different input ports only if those values were created concurrently; that is, at approximately the same time. However, values that were created at the same time may arrive at slightly different times due to computer speed or network traffic. This block considers two values to be concurrent if they arrive within a period of time you specify in the Concurrency Window attribute.

For example, suppose that Concurrency Window is 1 second. If you receive a value in the top input port and then a half second later receive a value in the bottom input port, the block considers the two values to be concurrent, stores them in the block's history as a pair, and passes a new value. However, if you receive a value in the bottom input port, and more than one second passes with no value received on the top input port, the block discards the value it received, and does not pass a new value.

If you receive several values over one input port before you receive a value over the other input port, the block discards all but the latest value it received.

Note If you are testing a diagram by entering data manually (for example, by using the override menu choice), make sure the Concurrency Window is large so you can enter values at a reasonable pace.

Specifying a Variable Instead of a Data Path

The Residual block can get one of its inputs from an attribute instead of a data path. To get the value from a path, set External Datasource to none. To get the value from an attribute, set External Datasource to one of the values in the following table, and the block ignores the input from the bottom data port (dp-in-2):

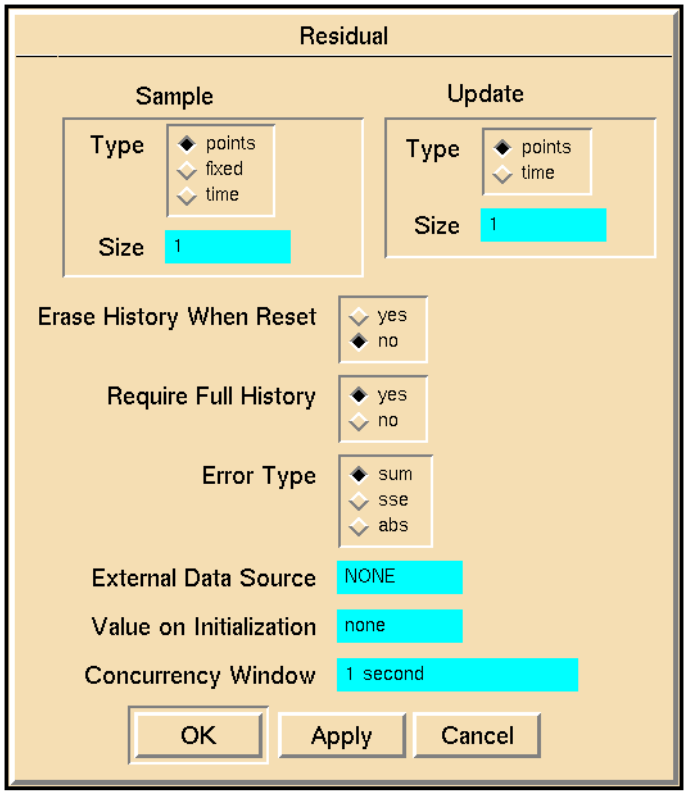
If External Datasource is…	The block…
A number	Uses that number as the value.
A symbol	Assumes the symbol names a G2 variable or parameter and gets the value from it.

If External Datasource is…	The block…
An embedded G2 variable	Gets the value from that variable or parameter. To create an embedded G2 variable, you must first go into Administrator mode, since the External Datasource attribute does not appear in the table in Developer, Browser, or User modes. Then, go to the block's attribute table, click on the attribute External Datasource, and select add optional subtable from the attribute's menu. When the External Datasource is given by an embedded variable, the External Datasource attribute in the configuration panel is grayed out, indicating that it has already been specified in the table.
A string	Assumes the string contains an expression that evaluates to a G2 variable or parameter. For more information, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.

Note If you set the External Datasource attribute, you may leave the bottom port (dp-in-2) unconnected, but you must always connect the top data port (dp-in-1).

Configuring

This is the configuration panel for the Residual block.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Error Type	Whether the block uses the difference between the top and bottom inputs to compute the error (sum), the absolute value of the difference (abs), or the square of the difference (sse).

Attribute	Description
External Datasource	Whether the block gets its bottom input from a path (none) or some other source, such as a variable.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.
Concurrency Window	The amount of time that the block applies to the top input value to be considered concurrent with the bottom input value.

Example

The following table shows some possible values for the Residual block, which has a history of 5 points. The first set of inputs has only negative differences. The second set of inputs has both positive and negative differences. The third set has some differences greater than 1.0 and some less than 1.0.

	The block passes this, when Error Type is…		
If the block receives these pairs of data…	sum	abs	sse
(3, 4) (1, 7) (5, 8) (2, 6) (7, 9)	-16	16	66
(3, 4) (7, 1) (5, 8) (6, 2) (7, 9)	4	16	66
(3, 4) (0.7, 0.1), (5, 8) (0.6, 0.2) (7, 9)	-5	7	14.52

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

8

Statistical Process Control

Describes the blocks that allow you to use statistical methods to measure process quality and consistency.

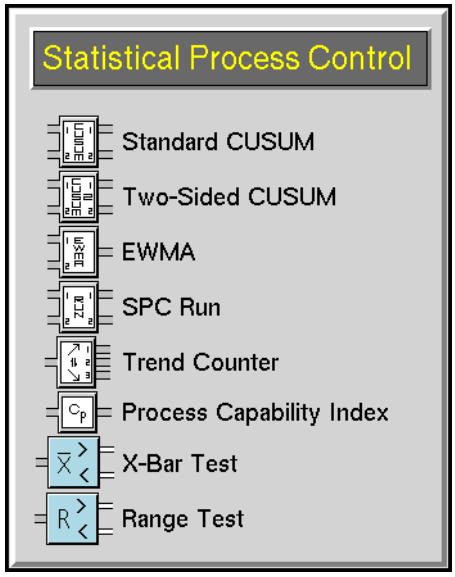
Introduction

SPC, or Statistical Process Control, is a way of using statistical methods to measure a process's quality and consistency. GDA has several blocks that help you use SPC techniques in your application.

An application that uses SPC techniques monitors the process's current performance and compares it to the expected performance. If the current performance differs significantly from the expected performance – that is, whether the process is out of control – GDA can issue an alarm. The operator can analyze the available information to find what caused the alarm and bring the

process back under control. The blocks in the SPC palette provide you with statistical tests, pattern recognition techniques, and graphs that let your application determine when a process is out of control and when to present that information to the operator.

You can find the Statistical Process Control palette under the Data submenu of the Palettes menu:



Specifying an Expected Value

Several of the blocks on this palette compare a measured value against an expected value. These blocks let you get the expected value from an input path or from an attribute. To get the value from a path, set the block's Expected Value attribute to none and connect the bottom input port (dp-in-2). To get the value from an attribute, set Expected Value to one of the values in this table, and the block ignores the input from the bottom input port (dp-in-2):

If Expected Value is…	The block…
A number	Uses that number as the expected value.
A symbol	Assumes the symbol names a G2 variable or parameter and gets the value from it.

If Expected Value is…	The block…
A string	Assumes the string contains an expression that evaluates to a G2 variable or parameter. For more information, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.
An embedded G2 variable or parameter	Gets the value from that variable or parameter. To create an embedded G2 variable, you must first go into Administrator mode, since by default, the Expected Value attribute does not appear in the table. Then, go to the block's attribute table, click on the attribute Expected Value select Add-optional-subtable from the attribute's menu. When the Expected Value is given by an embedded variable, the Expected Value attribute in the configuration panel is grayed out, indicating that it has already been specified in the table.

Note If you set the Expected Value attribute, you may leave the bottom port (dp-in-2) unconnected, but you must always connect the top data port (dp-in-1).

Accumulating Differences

The SPC accumulator blocks accumulate differences between the measured value and the expected value. The measured value is the data from your process. The expected value is what the data would be if the process were working perfectly. You pass in the measured values on the block's top input port. You can specify the expected values in one of two ways: set the attribute Expected Value to it, or pass it on the block's bottom input port. These are the accumulators:

- The Standard CUSUM block on page 194 measures trends above and below the expected value, and lets you specify a range of acceptable values with the attribute Slack.
- The Two-Sided CUSUM block on page 197 measures trends above and below an expected value, and lets you specify a range of acceptable values with two attributes: Upper Slack and Lower Slack.
- The EWMA block on page 200 measures trends away from the expected value and filters its output with an exponential filter.

- The SPC Run block on page 203 counts the number of successive values that are above or below the expected value.

Counting Trends

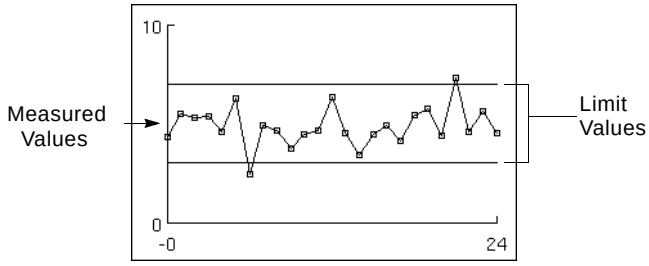
The Trend Counter block on page 205 detects patterns in your data. It counts the number of increasing, decreasing, and alternating trends in your data.

Computing the Process Capability Index

The Process Capability Index block on page 208 computes the process capability index (C_p) of the history of its input value. Like the Time Series blocks, described in "Time Series" on page 153, it maintains a history of values and operates on that history.

Creating Shewhart Charts

These blocks help you create Shewhart charts, which displays the measured values and the range of expected values, as this figure shows. They let you quickly determine when your data has exceeded its limits.



- The X-Bar Test block on page 211 displays a chart of the moving average of its input data.
- The Range Test block on page 214 displays a chart of the moving range of its input data.

Both of these blocks are Encapsulators, GDA blocks that contain diagrams on their subworkspaces. To configure them, you must go to their subworkspaces and configure the blocks contained on them. For more information on Encapsulators, see "Encapsulation" on page 477.

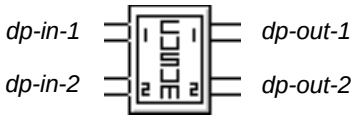
You can create your own Encapsulator blocks to create Shewhart charts of other properties of your data. See "Time Series" on page 153 for blocks that can perform useful analysis of your data.

See Also

Here are some other blocks that perform similar calculations:

- The Moving Average block on page 165 passes the average of the history of its input values.
- The Moving Range block on page 167 passes on the maximum, minimum, and range (the difference between the maximum and the minimum) of the block's history of values.
- The Variance block on page 159 passes either the statistical variance or the standard deviation of the history of its input value.
- The Statistical Moment block on page 178 calculates the deviation from a normal distribution of the block's history of values.
- The High Pattern and Low Pattern blocks on page 232 test whether a certain number of values from a history of values are above or below a threshold.

Standard CUSUM



The Standard CUSUM block measures trends above and below an expected value, and lets you specify a range of acceptable values with the attribute Slack. It expects its input data to be normally distributed.

The Standard CUSUM block passes these values:

- The top output port (dp-out-1) measures how far the measured values have been above the range. It is above zero when measured values have been above the range. It is near zero when measured values have been within the range. It never goes below zero.
- The bottom output port (dp-out-2) measures how far the measured values have been below the range. It is below zero when measured values have below the range. It is near zero when measured values have been within the range. It never goes above zero.

Pass the measured value on the top input port (dp-in-1). Set Slack to one-half the difference between the expected value and the closest value that is significantly different. If you do not specify a value for Slack, the block uses zero. Specify the expected value in one of two ways: with the bottom input port (dp-in-2) or with the attribute Expected Value, as described in “Specifying an Expected Value” on page 190.

Computing the Running Sums

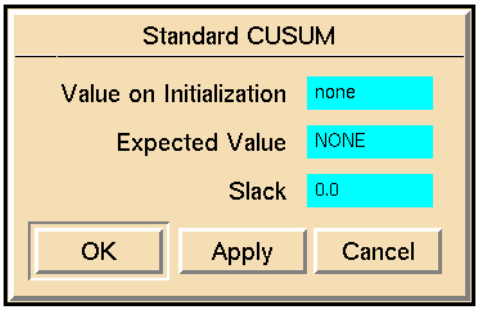
The block computes the difference between the measured value and the range of acceptable values and maintains two sums of the differences. The block passes the running sum of the net positive difference on the top output port (dp-out-1) and the running sum of the net negative difference on the bottom output port (dp-out-2).

$$pos\text{-}diff_t = \max(0, pos\text{-}diff_{t-1} + m - (e + Slack))$$

$$neg\text{-}diff_t = \min(0, neg\text{-}diff_{t-1} + m - (e - Slack))$$

Configuring

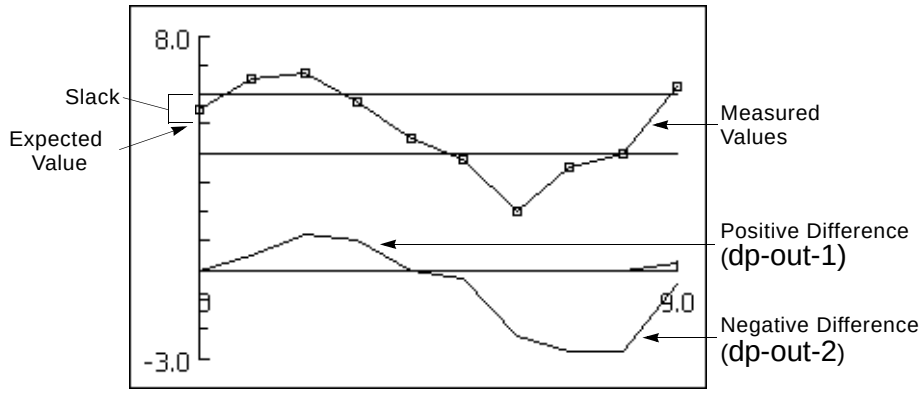
This is the configuration panel for the Standard CUSUM block.



Attribute	Description
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.
Expected Value	See "Specifying an Expected Value" on page 190.
Slack	The amount of uncertainty that the block applies to the input values to determine equivalency. Slack represents the uncertainty above or below the expected value; it is not an equivalence band.

Example

The following figure shows a chart of the output of a Standard CUSUM block, with an Expected Value of 5 and a Slack of 1. The top half of the graph shows the measured values. The two straight lines show the range of the acceptable values around the expected value.



This table shows the input and output values graphed in the previous figure. Notice when the measured value moves closer to the expected value (from 2.0 to 3.5), the bottom output still decreases since the output represents a cumulative sum of the differences. Also, notice that when the measured value shoots above the expected value (to 6.25) after being below it for a long time, both output values differ from 0.

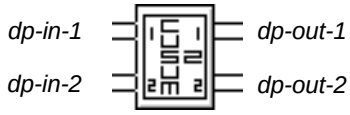
Measured Value	5.5	6.5	6.75	5.75	4.5	3.75	2.0	3.5	4.0	6.25
Top Output	0.0	0.5	1.25	1.0	0.0	0.0	0.0	0.0	0.0	0.25
Bottom Output	0.0	0.0	0.0	0.0	0.0	-0.25	-2.25	-2.75	-2.75	-0.5

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Two-Sided CUSUM



The Two-Sided CUSUM block measures trends above and below an expected value, and lets you specify a range of acceptable values around the expected value with the attributes Upper Slack and Lower Slack. It is especially useful when the input data is not normally distributed, such as when the data is auto-correlated.

The Two-Sided CUSUM block passes these values:

- The top output port (dp-out-1) measures how far the measured values have been above the range. It is above zero when measured values have been above the range. It is near zero when measured values have been within the range. It never goes below zero.
- The bottom output port (dp-out-2) measures how far the measured values have been below the range. It is below zero when measured values have been below the range. It is near zero when measured values have been within the range. It never goes above zero.

Pass the measured value on the top input port (dp-in-1). Set Upper Slack to one-half the difference between the expected value and the closest higher value that is significantly different. Set Lower Slack to one-half the difference between the expected value and the closest lower value that is significantly different. If you do not specify a value for Upper Slack or Lower Slack, the block uses zero. Specify the expected value in one of two ways: with the bottom input port (dp-in-2) or with the attribute Expected Value, as described in "Specifying an Expected Value" on page 190.

Computing the Running Sums

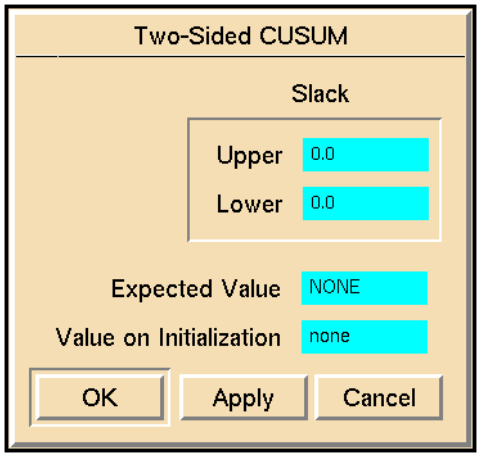
The block computes the difference between the measured value and the range of acceptable values and maintains two sums of the differences. The block passes the running sum of the net positive difference on the top output port (dp-out-1) and the running sum of the net negative difference on the bottom output port (dp-out-2).

$$pos\text{-}diff_t = \max(0, pos\text{-}diff_{t-1} + m - (e + Upper\text{-}slack))$$

$$neg\text{-}diff_t = \min(0, neg\text{-}diff_{t-1} + m - (e - Lower\text{-}slack))$$

Configuring

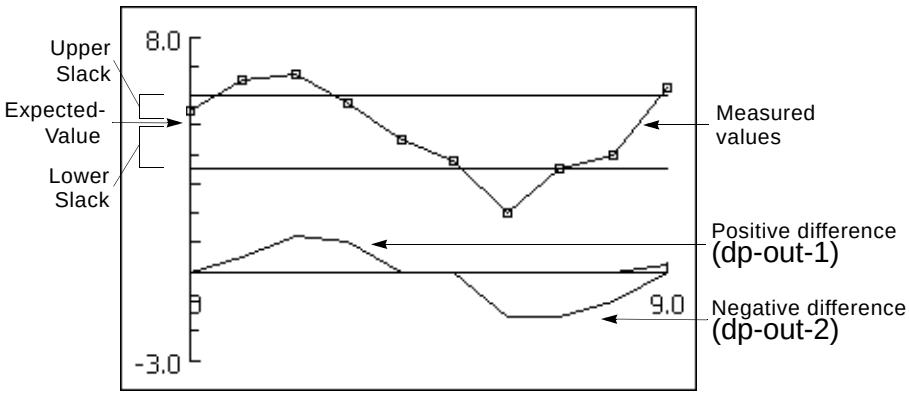
This is the configuration panel for the Two-Sided CUSUM block.



Attribute	Description
Upper Slack	The uncertainty above the expected value that the block applies to the input values to determine equivalency.
Lower Slack	The uncertainty below the expected value that the block applies to the input values to determine equivalency.
Expected Value	See “Specifying an Expected Value” on page 190.
Value on Initialization	See “Specifying an Initial Data Value” on page 83 in the *GDA User’s Guide*.

Example

This figure shows a chart of the output of a Two-Sided CUSUM block, with an Expected Value of 5, an Upper Slack of 1, and a Lower slack of 1.5. The top half of the graph shows the measured values. The two straight lines show the range of the acceptable values around the expected value.



This table shows the input and output values graphed in the previous figure:

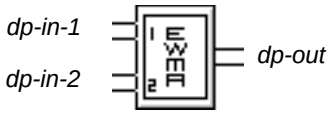
Measured Value	5.5	6.5	6.75	5.75	4.5	3.75	2.0	3.5	4.0	6.25
Top Output	0.0	0.5	1.25	1.0	0.0	0.0	0.0	0.0	0.0	0.25
Bottom Output	0.0	0.0	0.0	0.0	0.0	0.0	-1.5	-1.5	-1.0	0.0

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

EWMA



The EWMA (Exponentially Weighted Moving Average) measures trends beyond an expected value and filters these deviations with an exponential filter. It computes the output value by combining the previous output value and the difference between the current measured value and expected value. Specify the expected value in one of two ways: with the bottom input port (dp-in-2) or with the attribute Expected Value, as described in "Specifying an Expected Value" on page 190.

Specifying How to Filter

To specify how much the filter weights previous output values, set the attributes Filter Constant and Filter Mode. These attributes determine how the block computes the coefficient α, which must be between 0.0 and 1.0. The larger it is, the more weight the block gives to previous output. A common value for α is 0.8.

You must enter a positive number for the Filter Constant.

If you want the block to consider the amount of time between values when filtering (for example, give lesser weight to previous output if the last value was received 10 minutes ago than if it was received 10 seconds ago), set Filter Mode to time, and use the following equation to set Filter Constant so that it gives you the value you want for α. Note that the block must receive values at regular intervals if you use this method. If you want the block to ignore the amount of time between values when filtering (for example, give the same weight to previous output if the last value was received 10 minutes or 10 seconds ago), set Filter Mode to points and set Filter Constant to the value you want for α.

Computing the Value

The EWMA block uses this equation to compute its output value:

$$\text{output}_n = \alpha \cdot \text{output}_{n-1} + (1-\alpha)(\textit{measured-value} - \textit{expected-value})$$

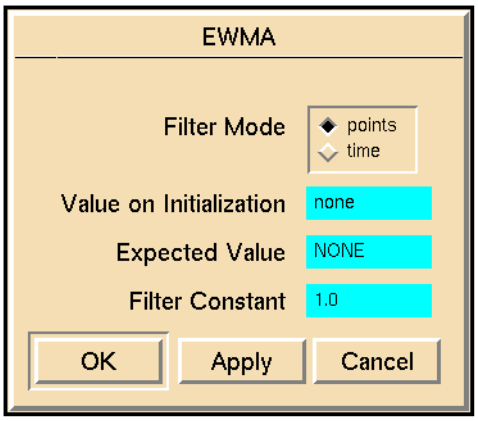
$$\textit{where } \alpha = \begin{cases} e^{-\frac{\Delta\text{time}}{\textit{Filter-constant}}}, & \text{if } \textit{Filter-mode} \text{ is time} \\ \textit{Filter-constant}, & \text{if } \textit{Filter-mode} \text{ is points} \end{cases}$$

The coefficient α is the amount of weight that the filter gives the current output, and the coefficient (1–α) is the amount of weight that the filter gives the previous output values.

When Filter Mode is time, the block computes α with both the Filter Constant and the difference in time between the last input and the current input. The constant α is computed such that values received long ago will not have as much effect on the current value as ones that were received a short time ago.

Configuring

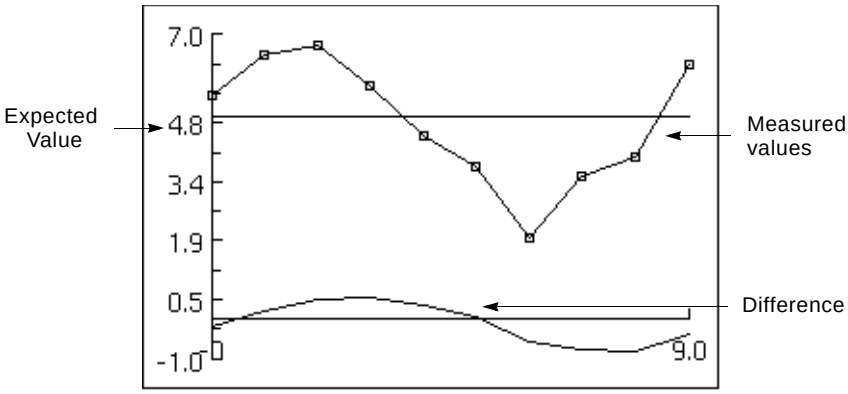
This is the configuration panel for the EMWA block.



Attribute	Description
Filter Mode	Whether the block considers the amount of time between values when filtering (time) or whether it ignores time.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.
Expected Value	See "Specifying an Expected Value" on page 190.
Filter Constant	The amount of weight that the filter gives the current output value over the previous output values, when Filter Mode is points, or the constant value that you use to compute this weight, when Filter Mode is time.

Example

This figure shows a chart of the output of a EWMA block, with an Expected Value of 5, a Filter Mode of points, and a Filter Constant of 0.8. The top half of the graph shows the measured values, and the bottom half shows the output values.



This table shows the input and output values graphed in the previous figure:

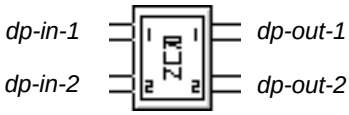
Measured Value	5.5	6.5	6.75	5.75	4.5	3.75	2.0	3.5	4.0	6.25
Output	-0.19	0.15	0.47	0.52	0.32	0.0	-0.6	-0.78	-0.82	-0.47

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Non-Linear Exponential Filter	page 78	*Reference Manual*

SPC Run



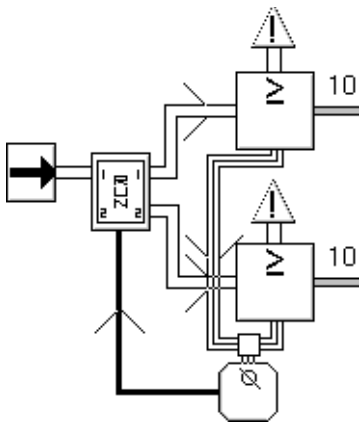
The SPC Run Accumulator counts the number of values in a row that are above or below the expected value. SPC Run passes these counts as shown in the following table:

The block passes the number of successive values…	On this port…
Over the expected value	The top output (dp-out-1)
Under the expected value	The bottom output (dp-out-2)

Specify the expected value in one of two ways with either the bottom input port (dp-in-2) or with the attribute Expected Value, as described in "Specifying an Expected Value" on page 190.

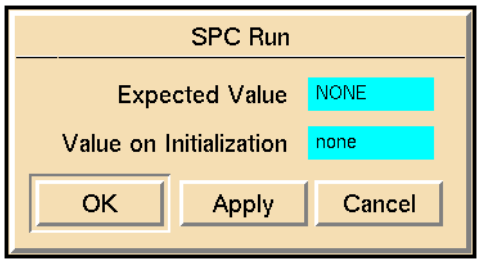
Usually, you will connect the output ports of SPC Run to High Value observations to detect trends above or below the expected values. You may also connect the High Value blocks to a Reset Block, which will reset the SPC Run whenever the observations detect a trend.

In this figure, the High Value blocks raise an alarm whenever there are more than 10 values over or under the expected value. Note that there is only one entry block connected to SPC Run, since the expected value is stored as an attribute of the block.



Configuring

This is the configuration panel for the SPC Run Accumulator.



Attribute	Description
Expected Value	See "Specifying an Expected Value" on page 190.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

If the Expected Value is 5 and the input values are 8, 7, 4, 6, 7, 8, SPC Run passes 3 on the top output and 0 on the bottom output, since the last three outputs (6, 7, and 8) are greater than the Expected Value.

If the Expected Value is 5 and the inputs are 9, 7, 8, 6, 3, SPC Run passes 0 on the top output and 1 on the bottom output, since the last output (3) is less than the Expected Value.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

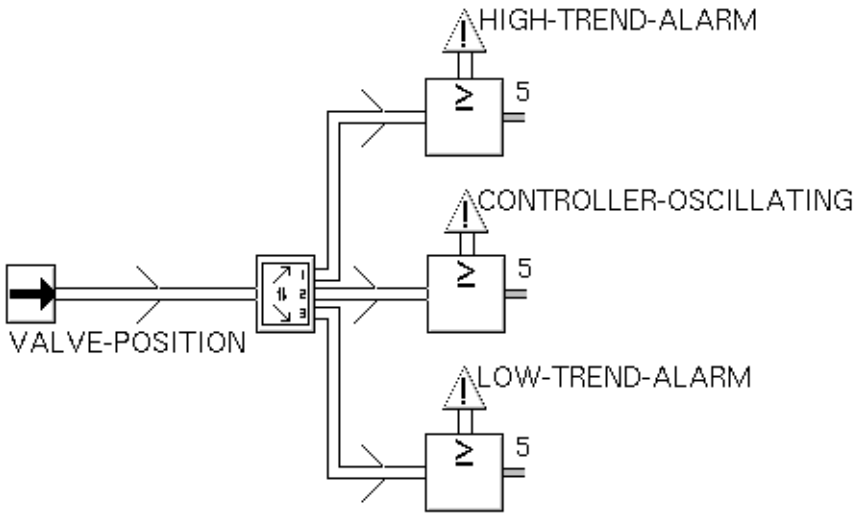
Trend Counter

The Trend Counter helps you detect increasing, decreasing, and alternating increasing and decreasing trends in data. This block is especially useful when you need to detect whether the data is random or not. The block passes the number of consecutive increasing, decreasing, and alternating values, this table shows:

The block passes the number of successive…	On this port…
Increasing values	Top port (dp-out-1)
Alternating increasing and decreasing values	Middle port (dp-out-2)
Decreasing values	Bottom port (dp-out-3)

The increasing and decreasing trends show you when the measured values are drifting away from a central value. You can attach High Value observations to the top and bottom ports to raise an alarm when the trend becomes significant.

Randomly sampled data should not successively alternate over a long period of time. You can attach a High Value observation to the middle port to raise an alarm when a sensor produces a significant number of alternating values. For example, suppose a controller turns a valve on or off depending on a sensor's value. If the controller keeps turning the valve on and off in rapid succession, there might be a problem. The following figure contains a Trend Counter connected to three High Value Observations, which raise alarms when there is a high trend, low trend, or a possible controller problem.

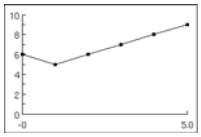


Configuring

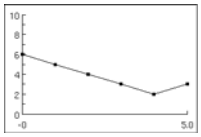
The Trend Counter has no configurable attributes.

Examples

If the input values are 6, 5, 6, 7, 8, and 9, as shown in this figure, Trend Counter passes 4 on its top port and 0 on the other two ports. The top port is 4 since the five values 5, 6, 7, 8, and 9 are a series of successively increasing values. The first value in the series (5) is not counted since it decreased from its previous value (6).

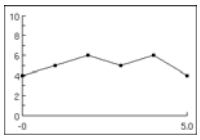


If the input values are 6, 5, 4, 3, 2, and 3, as shown in this figure, Trend Counter passes 1 on its top and middle ports and 0 on the bottom port. The top port is 0 and the bottom is 1 since the last value is increasing even though the first 5 values are successively decreasing values. The middle port is 1 since values switched once from decreasing to increasing.



If the input values are 4, 5, 6, 5, 6, and 4, as shown in this figure, Trend Counter passes 0 on the top port, 3 on the middle port and 1 on the bottom. The top port is 3, since the last three values (5, 6, and 4) have alternated between decreasing and

increasing. The bottom port is 1 since the last value (4) decreased from the previous value (6).

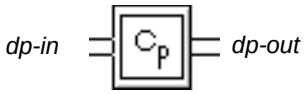


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Process Capability Index



The Process Capability Index block computes the process capability index (C_p) of the history of its input value. Once you set the Upper Specification Limit and the Lower Specification Limit correctly, C_P measures how well your process meets specifications. This table shows how C_P tells you the number of parts per million that are defective:

If C_P is…	Then this many parts per million are defective…
0.5	133,600
0.75	24,400
1.0	2,700
1.5	6.8
2.0	0.0018

The Process Capability Index uses this equation to figure C_P. (σ is the standard deviation for the history of values.)

$$C_P = \frac{\textit{Upper-specification-limit} - \textit{Lower-specification-limit}}{6\sigma}$$

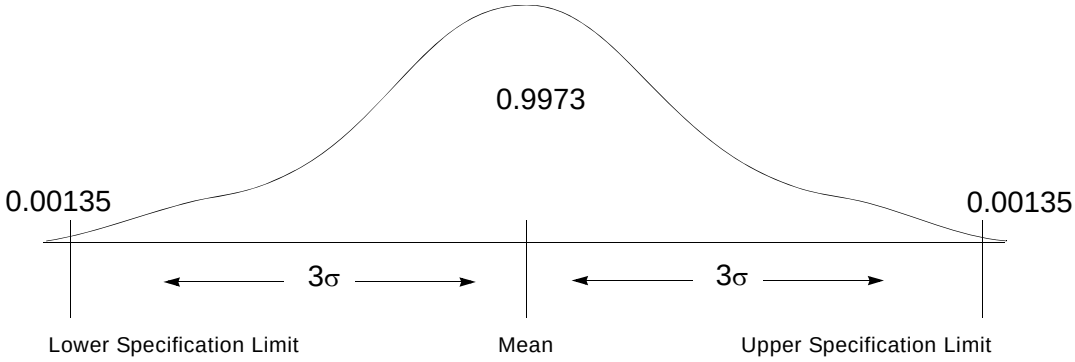
Like the blocks in "Time Series" on page 153, this block maintains a history of values and operates on that history.

Specifying a Sample Size

You must enter a value greater than 1 for the Sample Size attribute. The configuration panel prevents you from entering a value of less than 2.

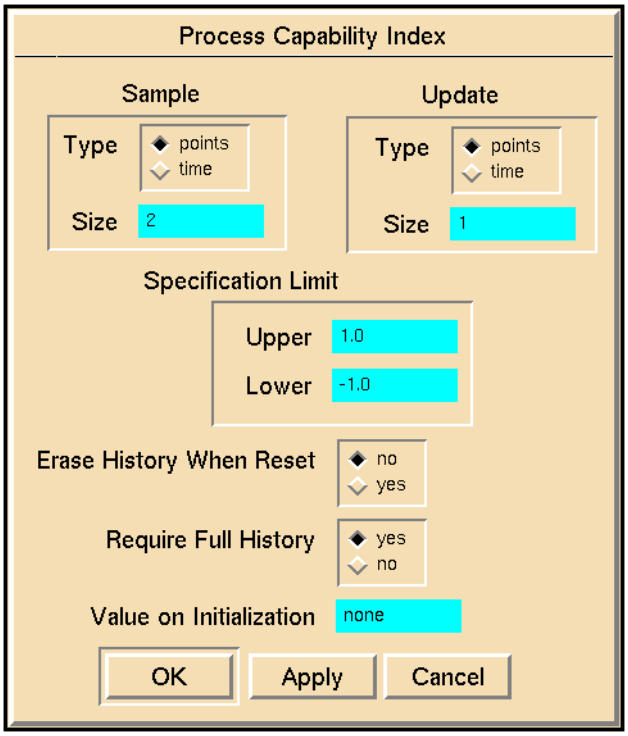
Specifying Upper and Lower Limits

Choose an Upper Specification Limit that falls at $\mu + 3\sigma$ and a Lower Specification Limit that falls at $\mu - 3\sigma$, where μ is the mean and σ is the standard deviation of the block's input value. If the input values have a normal distribution, 99.73% of the input values fall inside the limits and 0.27% fall outside. This figure shows an example:



Configuring

This is the configuration panel for the Process Capability Index block.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Update Type and Update Size	See "Specifying When to Propagate Data" on page 88 in the *GDA User's Guide*.

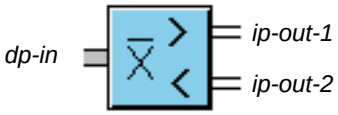
Attribute	Description
Upper Specification Limit	The mean of the block's input value plus three standard deviations.
Lower Specification Limit	The mean of the block's input value minus three standard deviations.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

See Also

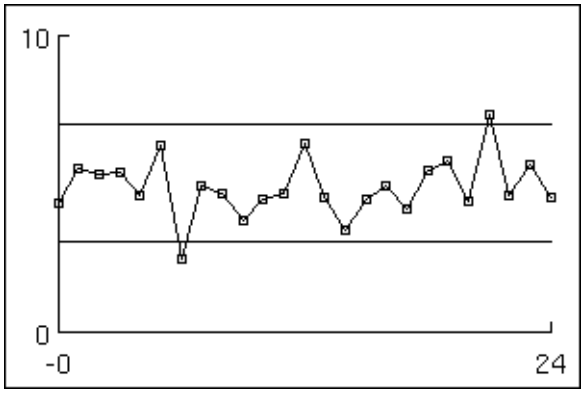
For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

X-Bar Test



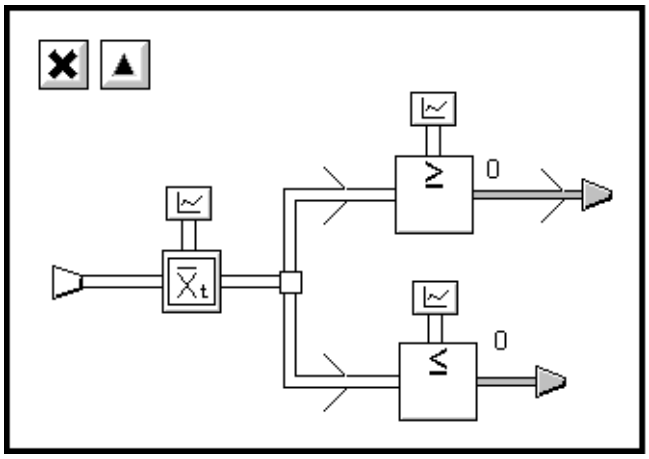
The X-Bar Test block passes on whether the moving average of its input data stays within a range of values, and displays a Shewhart chart of the moving average. A Shewhart chart, as shown in this figure, displays the upper and lower limits for the data, as well as the data:



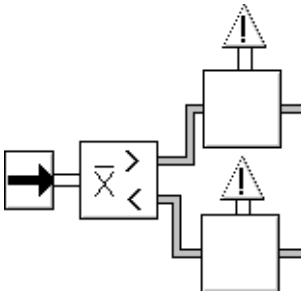
This block passes two inference values, shown in this table:

The block passes whether the moving average is…	On this port…
Above the upper limit	Top port (dp-out-1)
Below the lower limit	Bottom port (dp-out-2)

This block is an encapsulation block. To configure it, you must go to its subworkspace instead of its attribute table. Its subworkspace, in the following figure, contains a Moving Average block connected to a High Value and Low Value observation. Chart Capabilities are attached to each block and display the output of the Moving Average block, the Threshold of the High Value block, and the Threshold of the Low Value block.



You can connect the X-Bar Test block to Conditions, as in this figure, if you want to raise an alarm when the moving average falls outside the limits:



Setting Up the Block

To set up an X-Bar Test block:

1. Create and name a chart. You can create it with the New Display menu choice in the KB Workspace menu and name it with the Name menu choice in the chart's menu.
2. Go to the subworkspace of the X-Bar Test Block, by choosing view diagram from the block's menu.
3. In the tables of the three chart capabilities, set the attribute Chart-name to the name of the chart you created. The rest of the attributes are set properly.
4. Configure the Moving Average block on page 165.
5. Configure the High Value and Low Value blocks on page 223. Remember you can edit the attribute displays directly.

Configuring

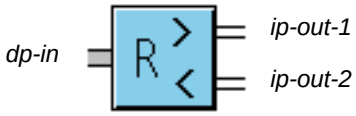
The X-Bar Test block is an encapsulation block; it has no configuration panel of its own. To configure an X-Bar Test block, choose view diagram from the block's menu, then configure the blocks on the subworkspace.

See Also

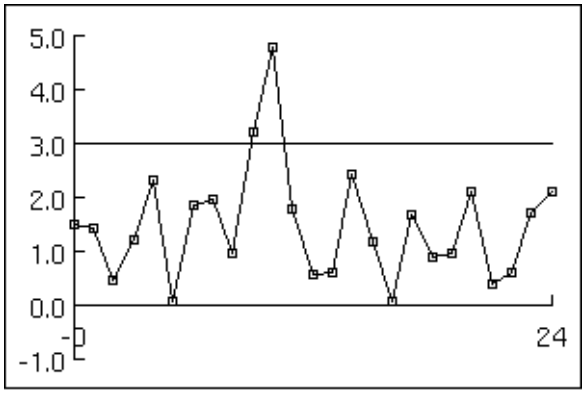
For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Range Test	page 214	*Reference Manual*
Encapsulation	page 477	*Reference Manual*

Range Test



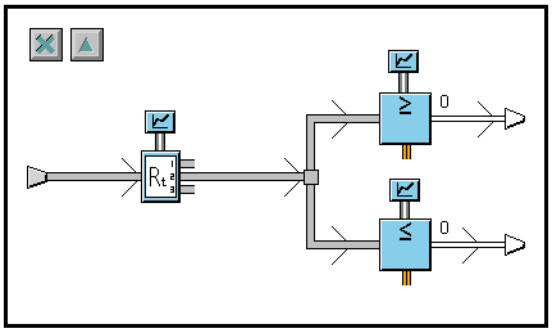
The Range Test block passes on whether the moving range of its input data stays within a range of values, and displays a Shewhart chart of the moving range. A Shewhart chart, as shown in this figure, displays the upper and lower limits for the data, as well as the data:



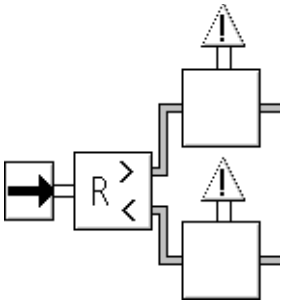
This block passes two values, shown in this table:

The block passes whether the moving range is…	On this port…
Above the upper limit	Top port (dp-out-1)
Below the lower limit	Bottom port (dp-out-2)

This block is an encapsulation block. To configure it, you must go to its subworkspace instead of its attribute table. Its subworkspace, in the following figure, contains a Moving Range block whose middle output, which carries the range, is connected to a High Value and Low Value observation. The top and bottom output ports, which carry the maximum and minimum values, are not used. Chart Capabilities are attached to each block and display the output of the Moving Range block, the Threshold of the High Value block, and the Threshold of the Low Value block.



You can connect the Range Test block to Conditions, as in this figure, if you want to raise an alarm when the range falls outside the limits:



Setting Up the Block

To set up an Range Test block:

1. Create and name a chart. You can create it with the New Display menu choice in the KB Workspace menu and name it with the Name menu choice in the chart's menu.
2. Go to the subworkspace of the Range Test Block, by choosing view diagram from the block's menu.
3. In the tables of the three chart capabilities, set the attribute Chart-name to the name of the chart you created. The rest of the attributes are set properly.
4. Configure the Moving Range block on page 167.
5. Configure the High Value and Low Value blocks on page 223. Remember you can edit the attribute displays directly.

Configuring

The Range Test block is an encapsulation block; it has no configuration panel of its own. To configure a Range Test block, choose view diagram from the block's menu, then configure the blocks on the subworkspace.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
X-Bar Test	page 211	*Reference Manual*
Encapsulation	page 477	*Reference Manual*

Part II

Inference Blocks

9

Observations

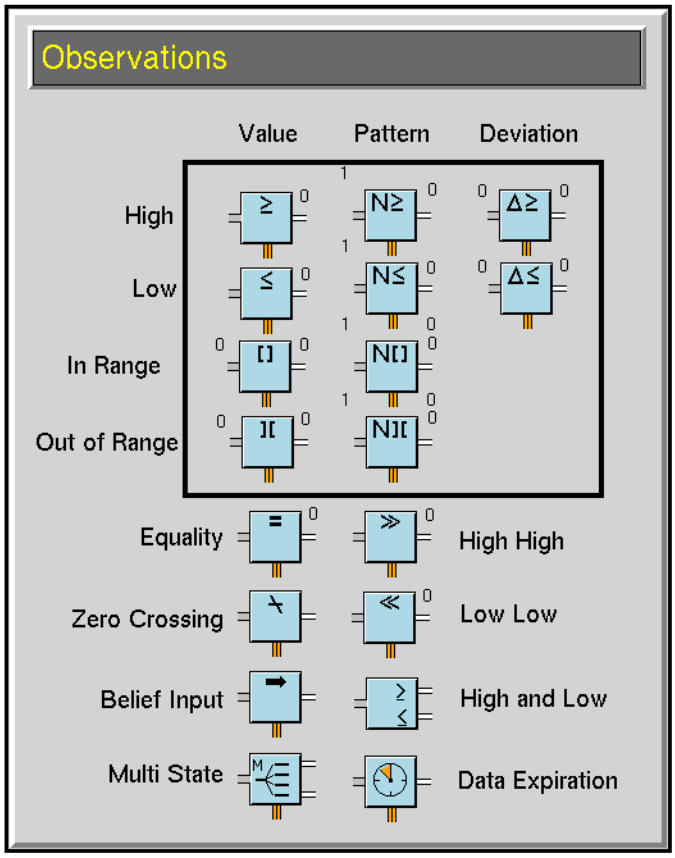
Describes blocks that detect features in data.

Introduction

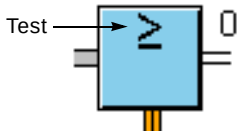
GDA includes a number of blocks that detect features in (or draw "observations" from) your data. These blocks take a data value as input, test it, and pass the inference value that the test produced as output. Because you can state the test as

an observation, such as “the input temperature is greater than or equal to 100°,” these blocks are called Observations. They mark the transition from the data stage to the inference stage of your diagram.

You can find Observations on the Observations palette under the Inference submenu of the Palettes menu:



All observations have similar icons. A symbol on the block indicates the test the block performs. This figure shows the High Value Observation. The symbol “≥” means the block performs the test greater-than-or-equal-to.



Note An input value with a Quality of no-value generates the output value unknown and a Quality of ok.

The Observations Palette

The blocks in the box at the top of the palette perform similar functions: they test their input values against thresholds. The column heading tells you what input values the block tests and the row heading tells you the condition the block is testing for, as shown in the following tables:

To test a…	Use a block in this column. . .
Single input value	Value
Certain number of values from the past	Pattern
Single input value for a condition that changes as your application runs	Deviation

To test whether a value is…	Use a block in this row. . .
Above a threshold	High
Below a threshold	Low
Inside a range	In Range
Outside a range	Out of Range

This chapter describes the blocks in the table column by column, then row by row. The High and Low blocks and the In Range and Out of Range blocks are described in the same sections. For example, the first three sections are “High Value, Low Value”, “In Range Value, Out of Range Value”, and “High Pattern, Low Pattern.”

Testing Against a Threshold

Several blocks test values against a threshold. These blocks include the blocks in the rows “High” and “Low” and three of the blocks below the table.

- To test a single input value against a threshold, use the High Value block and Low Value block on page 223 or the High High Value block and the Low Low Value block on page 256.
- To test a number of input values from the past against a threshold, use the High Pattern block or the Low Pattern block on page 232.

- To test a single input value against a threshold that changes as your application runs, use the High Deviation block or the Low Deviation block on page 243.
- To test whether a single input value is above one threshold and below another threshold, use the High and Low block on page 258. It passes two values.

The High High Value and Low Low Value blocks have different icons from the High Value and Low Value blocks, but have the same behavior and attributes. These blocks are useful when you need to distinguish between a slightly high (or low) value and a dangerously high (or low) value.

Testing a Range

Some blocks test values against a range. They include the blocks in the rows "In Range" and "Out of Range."

- To test whether a single input value is inside or outside a range, use the In Range Value block or the Out of Range Value block on page 227.
- To test whether a certain number of input values from the past are inside or outside the range, use the In Range Pattern block or the Out of Range Pattern block on page 237.

Using Alarms, Actions, and Explanations

The blocks on the Observations palette have special properties that most other inference blocks do not have. The properties include raising alarms, triggering action blocks, and producing explanations. To keep the inference blocks as simple as possible, these properties are restricted to the blocks on the Observations and Conditions palette. For more information see "Conditions" on page 375.

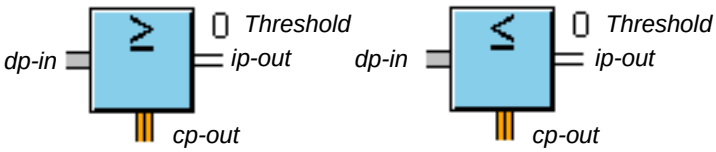
The menu choices current explanation, explanation of last true, explanation of last false, and explanation of last unknown appear in the menu for all observation blocks when an Explanation Memory capability is attached to the block. See "Explanation Memory" on page 533 for an explanation of these menu choices.

See Also

The blocks below the box on the Observations palette perform miscellaneous functions that test data values or convert them into belief values.

- To test whether the input value is equal to a reference value, use the Equality block on page 248.
- To test whether the current and previous input data values have different signs, use the Zero Crossing block on page 251.
- To convert a data value directly into a belief value, use the Belief Input block on page 253.

High Value, Low Value



The High Value block and the Low Value block test whether an input value is above or below the Threshold attribute. The High Value block tests whether the input value is greater than or equal to the Threshold. The Low Value block tests whether the input value is less than or equal to the Threshold.

GDA also includes the High High Value and Low Low Value blocks on page 256 to test for dangerous extremes.

Specifying a Threshold

Use the attributes Threshold and Threshold Uncertainty to specify a threshold.

Using Discrete Logic

To use discrete logic, set Threshold Uncertainty to none. The block compares the attribute Threshold against the input value.

- In a High Value block, the comparison returns .true if the value is greater than or equal to the Threshold and returns .false otherwise.
- In a Low Value block, the comparison returns .true if the value is less than or equal to the Threshold, and returns .false otherwise.

Using Fuzzy Logic

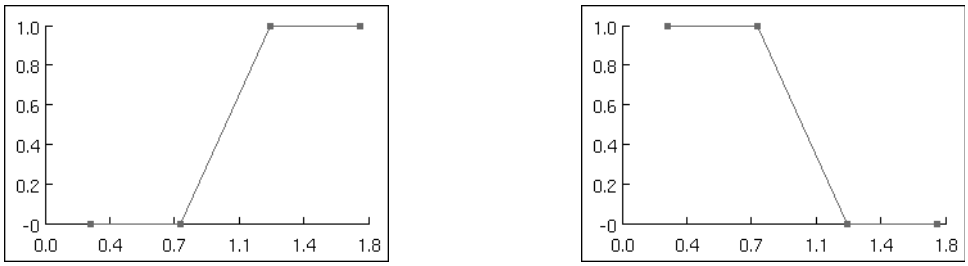
To use fuzzy logic, set both the attributes Threshold and Threshold Uncertainty. They specify a range of values that the block uses to scale the input value to a belief value, as in the following table. The blocks scale other values within the range accordingly.

If the input value is…	High Value returns…	Low Value returns…
Threshold + (Threshold Uncertainty/2) or larger	1.0	0.0
Threshold	0.5	0.5
Threshold – (Threshold Uncertainty/2) or smaller	0.0	1.0

Showing Membership Functions

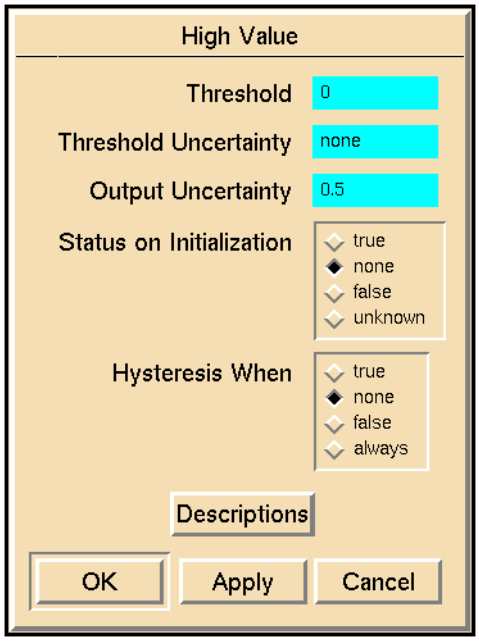
When you specify Threshold Uncertainty, GDA adds the choice show membership function to the block's menu. This choice displays a graph of the belief values for the input values ranging from Threshold – (Threshold Uncertainty / 2) to Threshold + (Threshold Uncertainty / 2). The X axis displays the input values, and the Y axis displays the belief values.

The following figure shows two sample membership function graphs: the left is for a High Value block and the right is for a Low Value block. Both blocks have a Threshold of 1.0 and a Threshold Uncertainty of 0.5.



Configuring

This is the configuration panel for the High Value block. The panel for the Low Value block is identical except for the block name.



Attribute	Description
Threshold	The value against which the block compares its inputs to determine the output inference value.
Threshold Uncertainty	An uncertainty band around the Threshold value that the block uses to determine whether an input value is high or low.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

Example

If both a High threshold and a Low threshold block have a Threshold of 75 and a Threshold Uncertainty of 50, the following table specifies the belief values they would pass for some sample input values:

If the input value is…	The high comparison returns…	The low comparison returns…
109	1.0	0.0
100	1.0	0.0
90	0.8	0.3
75	0.5	0.5
60	0.3	0.8

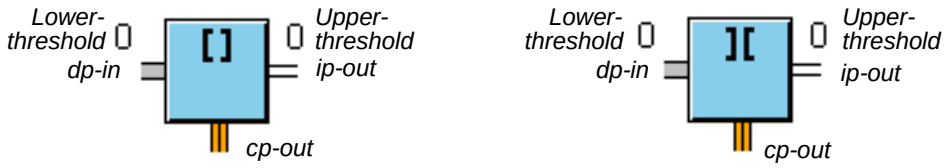
If the input value is…	The high comparison returns…	The low comparison returns…
50	0.0	1.0
41	0.0	1.0

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*

In Range Value, Out of Range Value



The In Range Value block and Out of Range Value block test whether an input value is inside or outside a range, which you specify with the attributes Lower Threshold and Upper Threshold.

The In Range block passes .true if the value is inside the range. The Out of Range block passes .true if the value is outside the range.

Specifying a Range

Use the attributes Lower Threshold, Upper Threshold, Lower Threshold Uncertainty, and Upper Threshold Uncertainty to specify the range to test for.

Using Discrete Logic

To use discrete logic, make sure the attributes Lower Threshold Uncertainty and Upper Threshold Uncertainty are set to none. The block tests whether an input value is inside or outside the range defined by the attributes Lower Threshold and Upper Threshold.

- In an In Range block, the comparison returns .true if the value is inside the range (that is, greater than or equal to the Lower Threshold and less than or equal to the Upper Threshold) and returns .false otherwise.
- In an Out of Range block, the comparison returns .true if the value is outside the range (that is, less than or equal to the Lower Threshold and greater than or equal to the Upper Threshold) and returns .false otherwise.

Using Fuzzy Logic

To use fuzzy logic, specify a Lower Threshold Uncertainty and Upper Threshold Uncertainty, in addition to a Lower Threshold and Upper Threshold. If the value is near either end of the range (that is, within a threshold uncertainty of a threshold), the comparison returns a belief value that is scaled between 0 and 1. If the value is well inside the range or outside the range, the comparison returns the belief value 0 or 1.

The following table shows the belief values that the comparisons return for an In Range and Out of Range block:

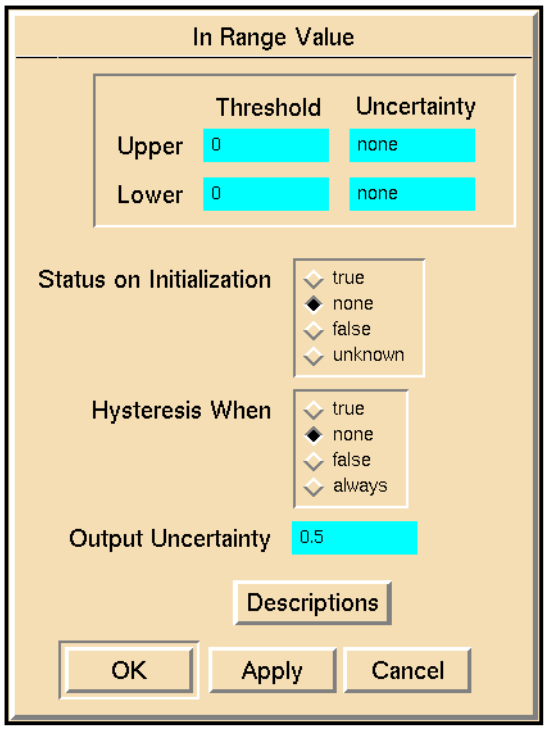
If the input value is…	In Range returns…	Out of Range returns…
Upper Threshold + (Upper Threshold Uncertainty/2) or larger	0.0	1.0
Upper Threshold	0.5	0.5
within Upper Threshold – (Upper Threshold Uncertainty/2) to Lower Threshold + (Lower Threshold Uncertainty/2)	1.0	0.0
Lower Threshold	0.5	0.5
Lower Threshold – (Lower Threshold Uncertainty/2) or smaller	0.0	1.0

Showing Membership Functions

When you specify an Upper Threshold Uncertainty or a Lower Threshold Uncertainty, GDA adds the choice show membership function to the block's menu. This choice displays a graph of the belief values for the input values ranging from Lower Threshold – (Lower Threshold Uncertainty/2) to Upper Threshold + (Upper Threshold Uncertainty/2). The X axis displays the input values and the Y axis displays the belief values. The figure in the following example shows two sample membership function graphs.

Configuring

This is the configuration panel for the In Range Value block. The panel for the Out of Range Value block is identical except for the block name.



Attribute	Description
Upper Threshold	The upper end of the range in which the block tests its inputs to determine the output inference value. The Upper Threshold must be greater than the Lower Threshold.
Upper Threshold Uncertainty	An uncertainty band around the Upper Threshold value that the block uses to determine whether an input value is in range or out of range.
Lower Threshold	The lower end of the range in which the block tests its inputs to determine the output inference value. The Lower Threshold must be less than the Upper Threshold.

Attribute	Description
Lower Threshold Uncertainty	An uncertainty band around the Lower Threshold value that the block uses to determine whether an input value is in range or out of range.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

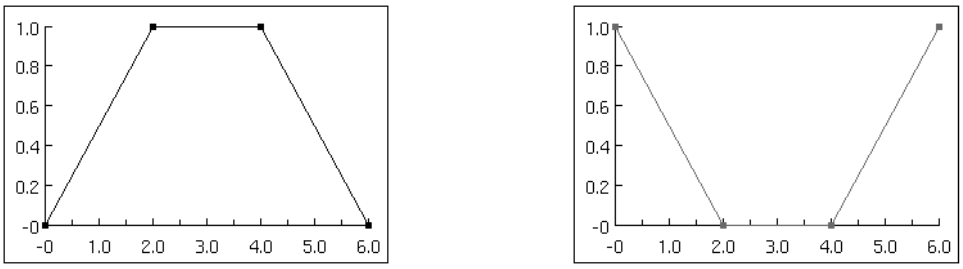
Example

If both an In Range Value block and an Out of Range Value block have a Lower Threshold of 1, an Upper Threshold of 5, and an Upper Threshold Uncertainty and a Lower Threshold Uncertainty of 2, the following table specifies the belief values that the range comparisons would return for some sample input values:

If the input value is…	In Range returns…	Out of Range returns…
-1.0	0.0	1.0
0.0	0.0	1.0
0.5	0.25	0.75
1.0	0.5	0.5
1.5	0.75	0.25
2.0	1.0	0.0
3.0	1.0	0.0
4.0	1.0	0.0

If the input value is…	In Range returns…	Out of Range returns…
4.5	0.75	0.25
5.0	0.5	0.5
5.5	0.25	0.75
6.0	0.0	1.0
7.0	0.0	1.0

This figure shows how those belief values look in a graph:

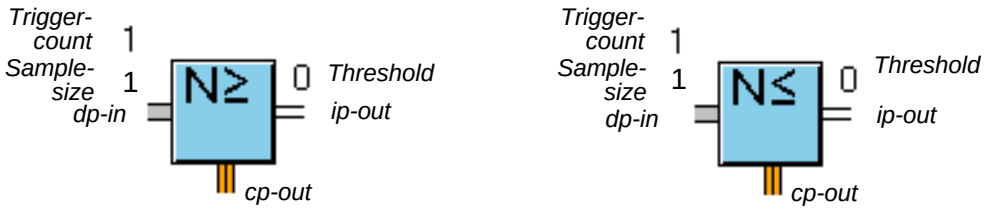


See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*
The Belief Range block	page 338	*Reference Manual*

High Pattern, Low Pattern



The High Pattern block and Low Pattern block test whether a certain number of values from a history of values are above or below a Threshold attribute. The number of values that must satisfy the condition is called the Trigger Count.

Note For more information on maintaining a history of values, see "Maintaining a History of Values" on page 84 in the *GDA User's Guide*.

The High Pattern block passes .true if Trigger Count values from the history are above the Threshold. The Low Pattern block passes .true if Trigger Count values from the history of values are below the Threshold.

The blocks pass the belief value that is the Trigger Count highest. For example, if the Trigger Count is 3, the blocks pass the third highest belief value from the history.

If Sample Type is points and Trigger Count is greater than Sample Size, the block sets Sample Size to the value of Trigger Count.

If the input Data-value has a Quality of no-value, then the block does nothing.

Specifying a Threshold

Use the attributes Threshold and Threshold Uncertainty to specify a threshold.

Using Discrete Logic

To use discrete logic, make sure the attribute Threshold Uncertainty is set to none. The block compares the attribute Threshold against another value.

- In a High Pattern block, the comparison returns .true if the value is greater than or equal to the Threshold and returns .false otherwise.
- In a Low Pattern block, the comparison returns .true if the value is less than or equal to the Threshold, and returns .false otherwise.

Using Fuzzy Logic

To use fuzzy logic, set the attributes Threshold and the Threshold Uncertainty attributes. They define a range of values that the block uses to scale the input

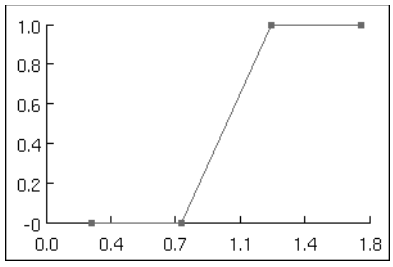
value to a belief value, according to the following table. The block scales other values within the range accordingly.

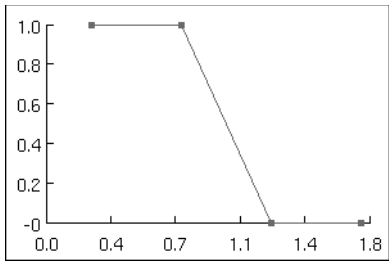
If the input value is…	High Value returns…	Low Value returns…
Threshold + (Threshold Uncertainty/2) or larger	1.0	0.0
Threshold	0.5	0.5
Threshold – (Threshold Uncertainty/2) or smaller	0.0	1.0

Showing Membership Functions

When you specify Threshold Uncertainty, GDA adds the choice show membership function to the block's menu. This choice displays a graph of the belief values for the input values ranging from Threshold – (Threshold Uncertainty/2) to Threshold + (Threshold Uncertainty/2). The X axis displays the input values, and the Y axis displays the belief values.

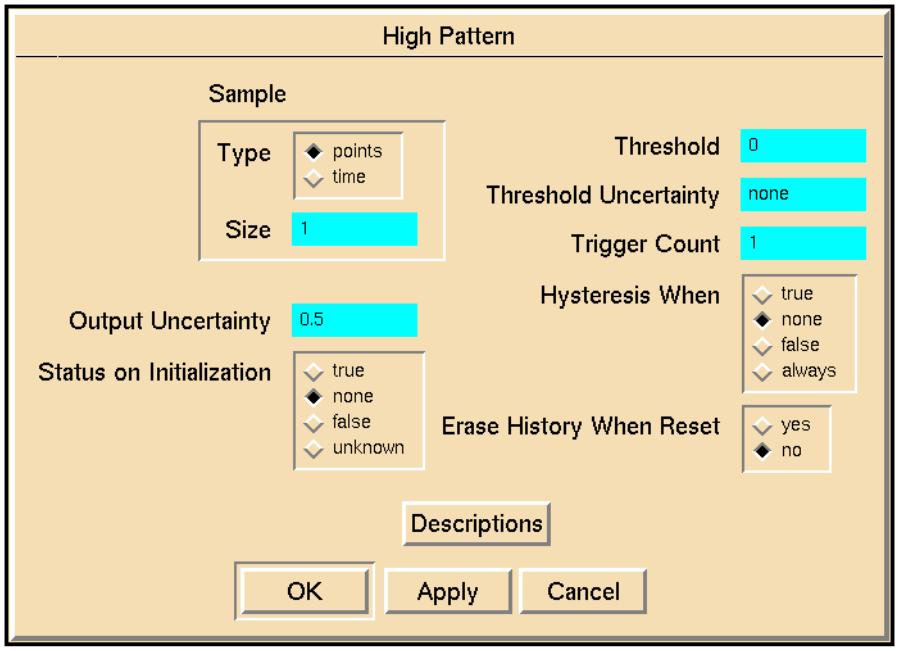
The following figure shows two sample membership function graphs: the left is for a High Value block and the right is for a Low Value block. Both blocks have a Threshold of 1.0 and a Threshold Uncertainty of 0.5.





Configuring

This is the configuration panel for the High Pattern block. The panel for the Low Pattern block is identical except for the block name.



Attribute	Description
Sample Type and Sample Size	See “Specifying the Size of the History” on page 85 in the *GDA User’s Guide*.
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Threshold	The value against which the block compares its inputs to determine the output inference value.
Threshold Uncertainty	An uncertainty band around the Threshold value that the block uses to determine whether an input value is high or low.
Trigger Count	The number of input values that must satisfy the condition that the block’s attributes describe.

Attribute	Description
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

Example

If both a High threshold and a Low threshold block have a Threshold of 75, a Threshold Uncertainty of 50, and a Trigger Count of 3, the following table shows the belief values they would pass for some sample values:

If the input value is…	The high comparison returns…	The low comparison returns…
109	1.0	0.0
100	1.0	0.0
90	0.8	0.2
85	0.7	0.3
75	0.5	0.5
65	0.7	0.3
60	0.2	0.8
50	0.0	1.0
41	0.0	1.0

Suppose the blocks' histories both contain 60, 85, 90, and 109. If you did not specify a Threshold Uncertainty, the High Pattern block would pass .true and the Low Pattern block would pass .false. Suppose you specify a Threshold Uncertainty of 50. For the High Pattern block, the belief values for the history are 0.2, 0.7, 0.8, and 1.0 and the block passes the belief value 0.8, the second highest

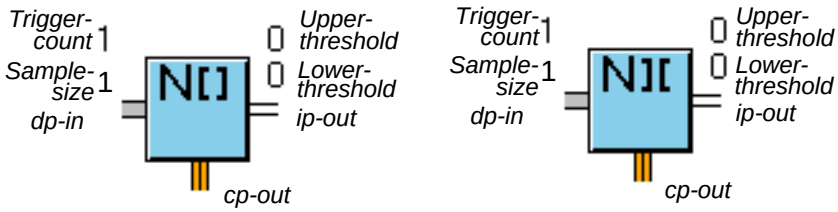
(since Trigger Count is 3). For the Low Pattern block, the belief values for the history are 0.8, 0.3, 0.2, and 0.0, and the block passes the belief value 0.2, the second lowest (since Trigger Count is 3).

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*

In Range Pattern, Out of Range Pattern



These blocks test whether a certain number of values from a history of values are inside or outside a range, which you specify with Lower Threshold and Upper Threshold. The number of values that must satisfy the condition is called the Trigger Count.

Note For more information on maintaining a history of values, see "Maintaining a History of Values" on page 84 in the *GDA User's Guide*.

The In Range Pattern block passes .true if Trigger Count values from the history are within the range. The Out of Range Pattern block passes .true if Trigger Count values from the history of values are outside the range.

The blocks output a belief value that is the Trigger Count highest. For example, if the Trigger Count is 3, the block's belief value is the third highest belief value from the history.

If Sample Type is points and Trigger Count is greater than Sample Size, the block sets Sample Size to the value of Trigger Count.

If the input Data-value has a Quality of no-value, then the block does nothing.

Specifying a Range

Use the attributes Lower Threshold, Upper Threshold, Lower Threshold Uncertainty, and Upper Threshold Uncertainty to specify the range to test for.

Using Discrete Logic

If you are using discrete logic, make sure the attributes Lower Threshold Uncertainty and Upper Threshold Uncertainty are set to none. A block tests whether the value is inside or outside the range defined by the attributes Lower Threshold and Upper Threshold.

- In an In Range block, the comparison returns .true if the value is inside the range (that is, greater than or equal to the Lower Threshold and less than or equal to the Upper Threshold) and returns .false otherwise.

- In an Out of Range block, the comparison returns .true if the value is outside the range (that is, less than or equal to the Lower Threshold and greater than or equal to the Upper Threshold) and returns .false otherwise.

Using Fuzzy Logic

If you are using fuzzy logic, specify the attributes Lower Threshold Uncertainty and Upper Threshold Uncertainty, in addition to a Lower Threshold and Upper Threshold. If the value is near either end of the range (that is, within the threshold uncertainty of a threshold), the comparison returns a belief value that is scaled between 0 and 1. If the value is well inside the range or outside the range, the comparison returns the belief value 0 or 1.

The following table shows the belief values that the comparisons return for an In Range and Out of Range block:

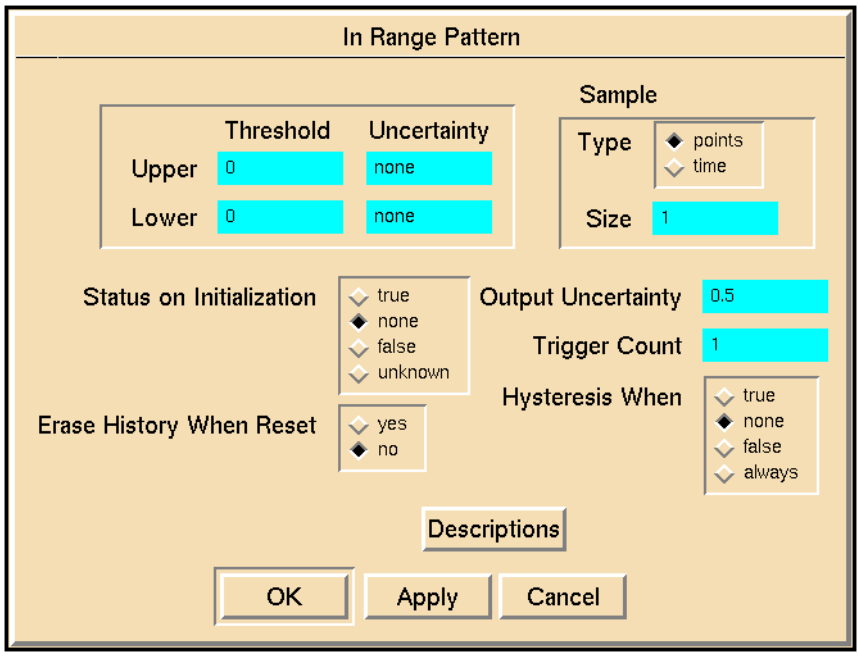
If the input value is…	In Range returns…	Out of Range returns…
Upper Threshold + (Upper Threshold Uncertainty/2) or larger	0.0	1.0
Upper Threshold	0.5	0.5
Upper Threshold – (Upper Threshold Uncertainty/2) to Lower Threshold + (Lower Threshold Uncertainty/2)	1.0	0.0
Lower Threshold	0.5	0.5
Lower Threshold – (Lower Threshold Uncertainty/2) or smaller	0.0	1.0

Showing Membership Functions

When you specify an Upper Threshold Uncertainty or a Lower Threshold Uncertainty, GDA adds the choice show membership function to the block's menu. This choice displays a graph of the belief values for the input values ranging from Lower Threshold – (Lower Threshold Uncertainty/2) to Upper Threshold + (Upper Threshold Uncertainty/2). The X axis displays the input values. The figure in the following example shows two sample membership function graphs and the Y axis displays the belief values.

Configuring

This is the configuration panel for the In Range Pattern block. The panel for the Out of Range Pattern block is identical except for the block name.



Attribute	Description
Upper Threshold	The upper end of the range in which the block tests its inputs to determine the output inference value. The Upper Threshold must be greater than the Lower Threshold.
Upper Threshold Uncertainty	An uncertainty band around the Upper Threshold value that the block uses to determine whether an input value is in range or out of range.
Lower Threshold	The lower end of the range in which the block tests its inputs to determine the output inference value. The Lower Threshold must be less than the Upper Threshold.
Lower Threshold Uncertainty	An uncertainty band around the Lower Threshold value that the block uses to determine whether an input value is in range or out of range.

Attribute	Description
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Trigger Count	The number of input values that must satisfy the condition that the block's attributes describe.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

Example

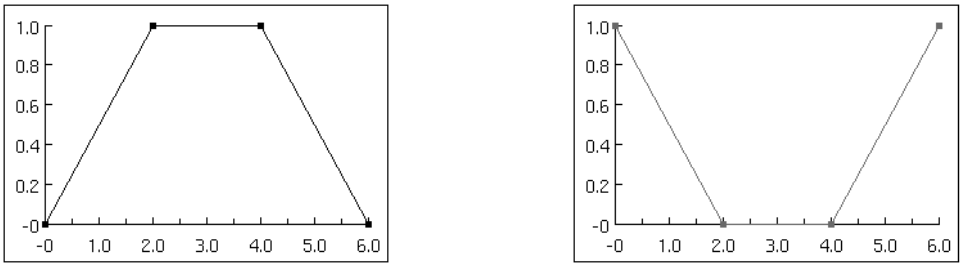
If both an In Range Pattern block and an Out of Range Pattern block have a Lower Threshold of 5, an Upper Threshold of 10, a Trigger Count of 3, and an Upper Threshold Uncertainty and a Lower Threshold Uncertainty of 2, the following table specifies the belief values that the range comparisons would return for some sample input values:

If the input value is…	In Range returns…	Out of Range returns…
-1.0	0.0	1.0
0.0	0.0	1.0
0.5	0.25	0.75
1.0	0.5	0.5
1.5	0.75	0.25

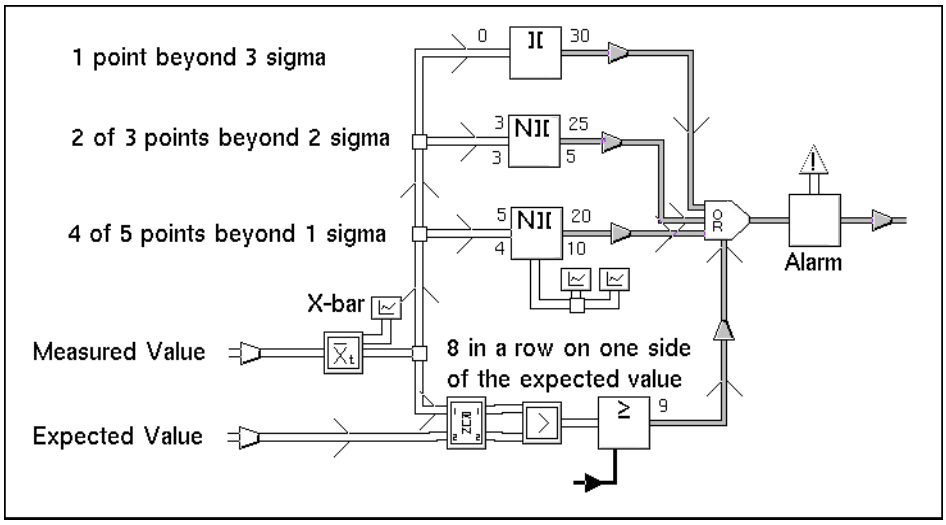
If the input value is…	In Range returns…	Out of Range returns…
2.0	1.0	0.0
3.0	1.0	0.0
4.0	1.0	0.0
4.5	0.75	0.25
5.0	0.5	0.5
5.5	0.25	0.75
6.0	0.0	1.0
7.0	0.0	1.0

Suppose the histories for the blocks both contain 4, 6, 8, 10. If the uncertainties are not specified, the In Range Pattern block passes .true and the Out of Range Pattern block passes .false. Suppose the blocks also have a upper and lower threshold uncertainties of 2. For the In Range Pattern block, the belief values for the history are 0.25, 0.75, 1.0, 0.5 and the block passes the belief value 0.5, the third highest (since Trigger Count is 3). For the Out of Range Pattern block, the belief values for the history are 0.75, 0.25, 0.0, 0.5, and the block passes the belief value 0.25, the third highest (since Trigger Count is 3).

This figure shows these belief values graphed against the values in the histories:



In this figure, a subworkspace implements the Western Electric rules, which test whether a measured value stays within set ranges of quality.

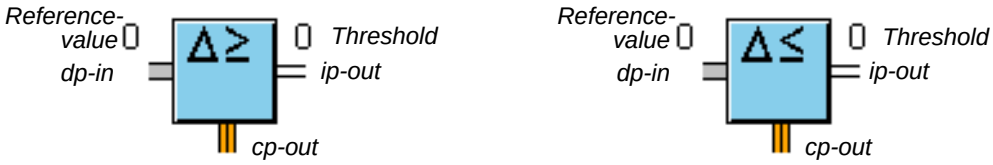


See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
The Belief Range block	page 338	*Reference Manual*

High Deviation, Low Deviation



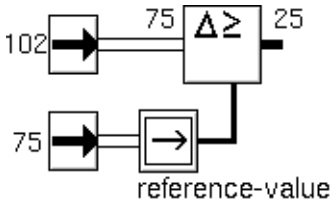
If you need to test whether a value is beyond a borderline that changes as your application runs, use the High Deviation and Low Deviation blocks. These blocks compute a borderline value by adding the attributes Threshold and Reference Value. By convention, the Threshold is fixed, and the Reference Value is set as your application is running.

Note Use the Set Attribute block on page 147 to change an attribute's value.

The High Deviation block passes .true if the input value is greater than or equal to Threshold + Reference Value, and passes .false otherwise. The Low Deviation block passes .true if the input value is less than or equal to Threshold + Reference Value, and passes .false otherwise.

For example, suppose you need to test whether a temperature is 25 degrees greater than a set point, but that set point varies as your application runs. Use a High Deviation block, and set its Threshold to 25. With a Set Attribute block, set its Reference Value to the set point as the set point changes.

In the following figure, the Set Attribute block is setting Reference Value to 75. Since the High Deviation block's input value is over the sum of the Reference Value and the Threshold (75 + 25 or 100), it passes .true.



Specifying a Threshold

Use the attributes Threshold, Threshold Uncertainty, and Reference Value to specify a threshold.

Using Discrete Logic

To use discrete logic, make sure the attribute Threshold Uncertainty is set to none. The block compares the sum of Threshold and Reference Value against another value.

- In a High threshold block, the comparison returns .true if the value is greater than or equal to that sum and returns .false otherwise.
- In a Low threshold block, the comparison returns .true if the value is less than or equal to that sum, and returns .false otherwise.

Using Fuzzy Logic

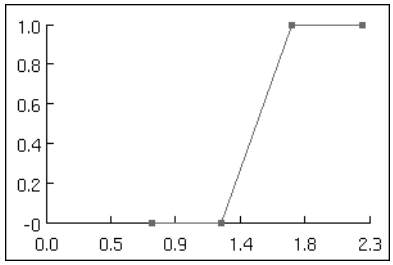
To use fuzzy logic, set the attribute Threshold Uncertainty, in addition to the attributes Threshold and Reference Value. They define a range of values that the block uses to scale the input value to a belief value, according to the following table. The block scales other values within the range accordingly.

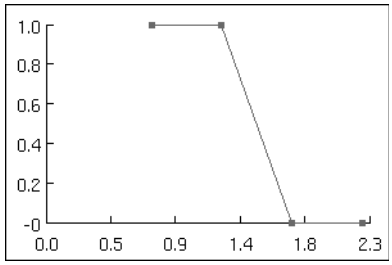
If the input value is…	The high comparison returns…	The low comparison returns…
Threshold + Reference Value + (Threshold Uncertainty/2) or larger	1.0	0.0
Threshold + Reference Value	0.5	0.5
Threshold + Reference Value – (Threshold Uncertainty/2) or smaller	0.0	1.0

Showing Membership Functions

When you specify Threshold Uncertainty, GDA adds the choice show membership function to the block's menu. This choice displays a graph of the belief values for the input values ranging from Threshold + Reference Value – (Threshold Uncertainty/2) to Threshold + Reference Value + (Threshold Uncertainty/2). The X axis displays the input values, and the Y axis displays the belief values.

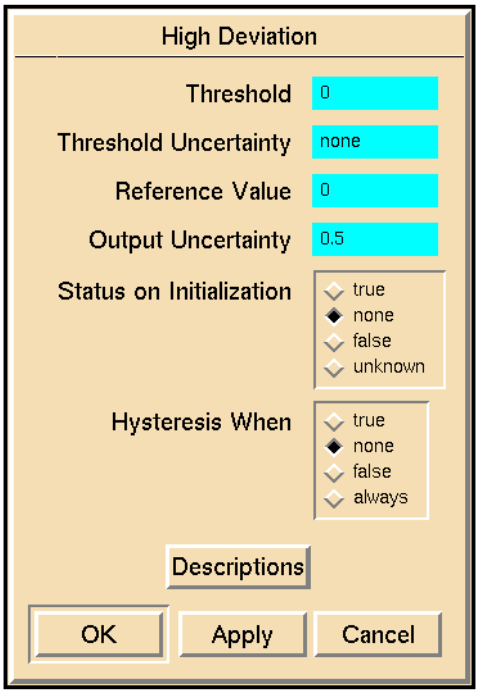
The following figure shows two sample membership function graphs: the left is for a High Value block and the right is for a Low Value block. Both blocks have a Threshold of 1.0, a Reference Value of 0.5, and a Threshold Uncertainty of 0.5.





Configuring

This is the configuration panel for the High Deviation block. The panel for the Low Deviation block is identical except for the block name.



Attribute	Description
Threshold	The value against which the block compares its inputs to determine the output inference value.
Threshold Uncertainty	An uncertainty band around the Threshold value that the block uses to determine whether an input value is high or low.
Reference Value	The value that the block adds to the Threshold value while the application is running to compute a dynamically changing threshold.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

Example

If both a High Deviation and a Low Deviation block have a Threshold of 50, a Reference Value of 25, and a Threshold Uncertainty of 50, the following table shows the belief values they would pass for some sample input values:

If the input value is…	The high comparison returns…	The low comparison returns…
109	1.0	0.0
100	1.0	0.0
90	0.8	0.3
75	0.5	0.5
60	0.3	0.8

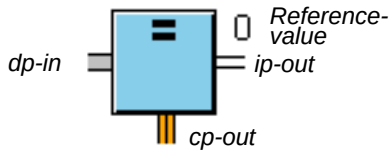
If the input value is…	The high comparison returns…	The low comparison returns…
50	0.0	1.0
41	0.0	1.0

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*

Equality

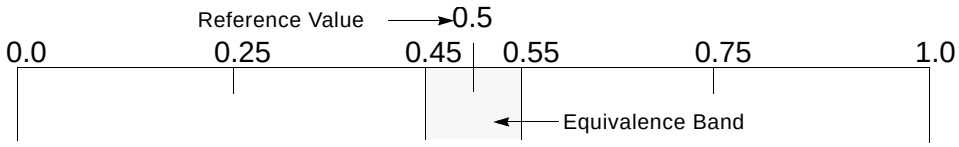


The Equality block checks whether an input value is equal to the attribute Reference Value. It passes .true if they are equal and .false otherwise. The Reference Value can either be a number, a symbol, or a text string, but its data type must match the data type of the input value.

If you want to check whether the input is approximately equal to the Reference Value, use the Equivalence Band attribute. When the input value is in the following range, the Equality block considers the values equal and passes .true:

Reference Value – Equivalence Band/2 ≤ input value <
Reference Value + Equivalence Band/2

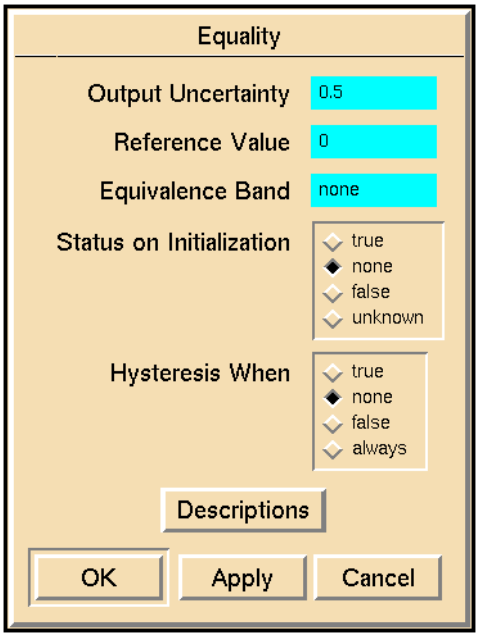
For example, if an Equality block has a Reference Value of 0.5 and an Equivalence Band of 0.1, as shown in the following figure, it passes .true for when the input value is less than 0.55 and greater than or equal to 0.45. This attribute is especially useful for comparing floating-point numbers, which can differ by small amounts due to rounding errors but still appear to be equal.



The Equality block performs only discrete logic, even if you supply an Equivalence Band. To perform fuzzy logic, use the In Range and Out of Range blocks on page 227.

Configuring

This is the configuration panel for the Equality block.



Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Reference Value	The value against which the block compares its inputs to determine equality.
Equivalence Band	The amount of uncertainty that the block applies to the input value to determine equivalency. The number represents an uncertainty band around the input value.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.

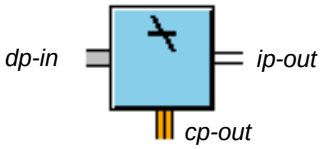
Attribute	Description
Hysteresis When	See “Specifying Hysteresis” on page 103 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Overriding Block Values	page 51	*User’s Guide*
Editing Attribute Displays	page 27	*User’s Guide*
Specifying and Generating Explanations	page 93	*User’s Guide*
Evaluating Expressions in Attributes	page 118	*User’s Guide*
Specifying Fuzzy Logic Attributes	page 101	*User’s Guide*

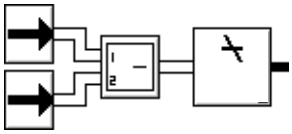
Zero Crossing



Use the Zero Crossing block to detect when a data value changes its sign.

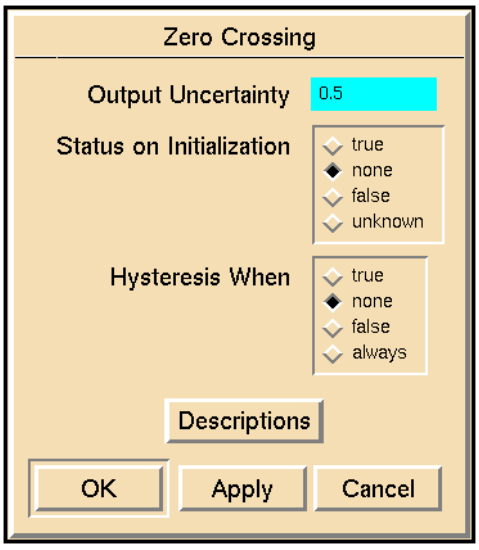
This block passes .true when the current input value has a different sign from the previous input value. For example, if the current input were -1.0 and the previous input were 0.005, it would pass .true. If the current input were 0.0 and the previous input were 0.5, it would pass .false.

If the input value is the difference between two signals, you can use this block to detect signal crossing events, as shown in this figure:



Configuring

This is the configuration panel for the Zero Crossing block.



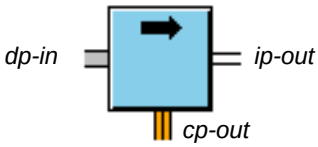
Attribute	Description
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*

Belief Input



The Belief Input block converts its input data value to a belief value.

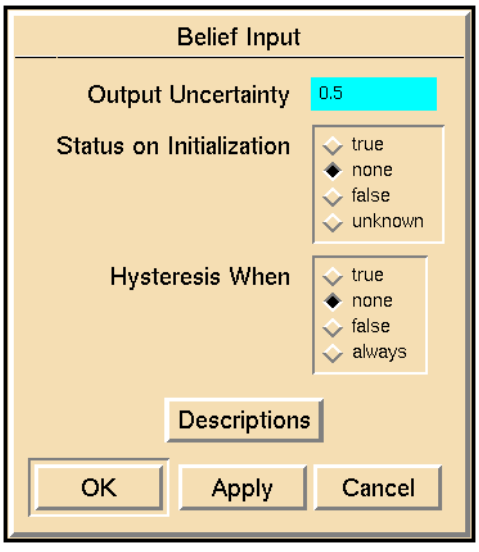
Since belief values must fall within the range 0.0 to 1.0, this block changes data values less than 0.0 to belief values of 0.0 and data values greater than 1.0 to belief values of 1.0. This block converts data values in the range 0.0 to 1.0 to equivalent belief values.

This block is frequently used with the Linear Scaling block. The Linear Scaling block scales its input data value to the range 0.0 to 1.0 and the Belief Input block converts that value to an inference value.

Note If the Quality of the input data path is no-value then the output inference path will have a Quality of ok, a Status-value of unknown, and a Belief-value of 0.5.

Configuring

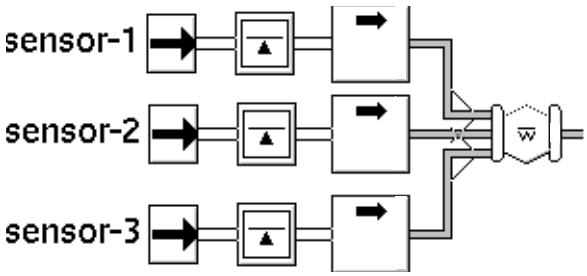
This is the configuration panel for the Belief Output block.



Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Hysteresis When	See “Specifying Hysteresis” on page 103 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

Example

In the following figure, three Linear Scaling blocks and Belief Input blocks convert data values from three entry points to belief values. The data values are from three sensors, which have values from 0 to 100, where 0 means the equipment is unlikely to fail and 100 means the equipment is likely to fail. The Weighted Evidence Combiner figures whether or not the three sensors agree that the equipment is likely to fail.



To convert these values to belief values, the Linear Scaler blocks are set up as shown in this table:

Attribute	Value
Input Lower Bound	0.0
Input Upper Bound	100.0

Attribute	Value
Output Lower Bound	0.0
Output Upper Bound	1.0

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*
The Belief Output block	page 340	*Reference Manual*

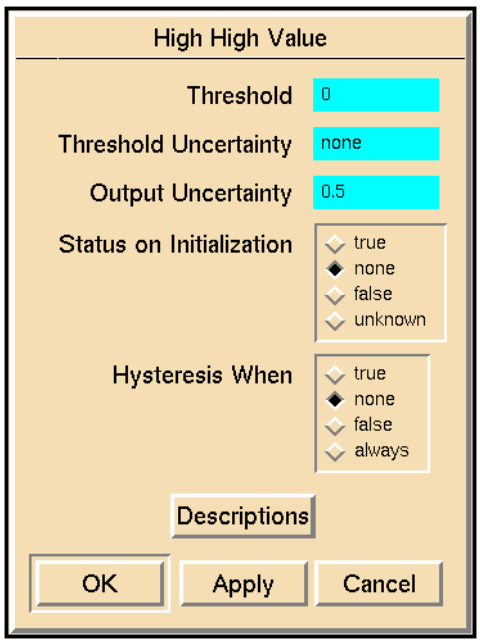
High High Value, Low Low Value

The High High Value and Low Low Value blocks have different icons from the High Value and Low Value blocks, but have the same behavior and attributes. They are useful when you need to distinguish between a slightly high (or low) value and a dangerously high (or low) value. For example, you may want to warn the operator if a tank's pressure is over 100psi and evacuate the facility if it is over 1000psi. By convention, use the High High Value and Low Low Value blocks to test for dangerous extremes only.

For more information, refer to "High Value, Low Value" on page 223.

Configuring

This is the configuration panel for the High High Value block. The panel for the Low Low Value block is identical except for the block name.



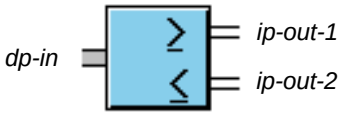
Attribute	Description
Threshold	The value against which the block compares its inputs to determine the output inference value.
Threshold Uncertainty	An uncertainty band around the Threshold value that the block uses to determine whether an input value is high or low.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

See Also

For more information on attributes and menu choices that are not described in this section, see the sections below.

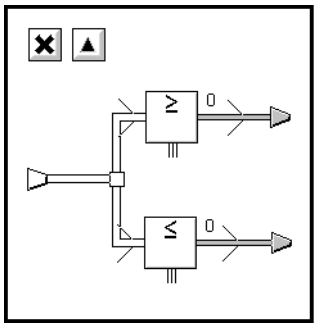
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Editing Attribute Displays	page 27	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*

High and Low



The High and Low block combines two tests into one block. The top path passes whether the input value is above a threshold. The bottom path passes whether the input value is below another threshold.

The High and Low block is an encapsulation block, which has a subworkspace with the diagram in the following figure. To configure the High and Low block, go to its subworkspace and set the Threshold attributes for the High Value and Low Value blocks.



For more information, see “Encapsulation Block” on page 479 and “High Value, Low Value” on page 223.

Configuring

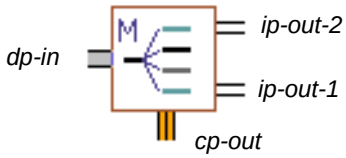
The High and Low block is an encapsulation block; it has no configuration panel of its own. To configure a High and Low block, choose view diagram from the block’s menu, then configure the blocks on the subworkspace.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*

Multi-State



The Multi-state block classifies an input into one of some number of categories, or states. You specify the number of output states by using the Number of States attribute.

You use this block in two distinct ways, depending on the specification of Input Type:

If Input Type is...	You specify...	And you specify...
symbolic or textual	The same number of symbolic Category values as the number of output states	Logic as discrete
numeric	One fewer numeric Threshold values than the number of output states (and Threshold Uncertainty when using fuzzy logic)	Logic as discrete or fuzzy

The Multi-state block sends a control signal on its control output path (cp-out) when the value of any output inference path turns to .true.

Changing the Number of States

The Number of States attribute determines the number of output inference paths on the Multi-state block, and thus the number of observation states.

To change the number of observation states:

1. Select configure from the block's menu.
2. Click on the arrows next to the box associated with Number of States.

   This automatically updates the number of attributes associated with each state displayed in the configuration panel.

There are one fewer Thresholds and Threshold Uncertainties attributes than the number of states, and the same number of Categories and Explanations attributes as the number of states.

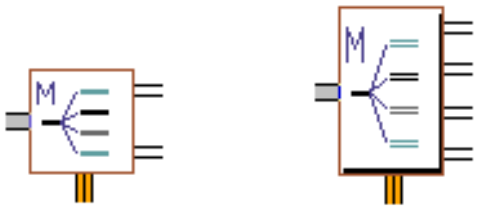
For example, if you change the Number of States from two to four, the number of Thresholds and Threshold Uncertainties attributes goes from one to three, and the number of Categories and Explanations attributes goes from two to four.

Use the scroll buttons to display attributes that do not fit on the screen. The scroller causes all of the lists to scroll simultaneously so that the attributes are synchronized.

When you select the OK button on the configuration panel, GDA reconfigures the block, adding or removing output ports as necessary. If you increase the number of states, the block maintains existing connections.

Note If the block is already connected to other blocks, do not decrease the number of states; otherwise, the behavior is unpredictable.

The following figure illustrates the result of changing the Multi-state block from two output states to four output states. Use the Change-size menu option to adjust the size of the block.



Number of States = 2 Number of States = 4

Note The numbering of the output ports for the Multi-state block goes from bottom to top. Thus, ip-out-1 is at the bottom of the block and ip-out-*n* is at the top. This numbering scheme corresponds to the ordering of the thresholds, which decrease as you go down the output paths.

Using Numeric Inputs

To use numeric inputs, specify the Input Type as numeric.

The following headings describe how you use numeric threshold values in both discrete and fuzzy modes.

Using Discrete Logic

You use discrete logic to test numeric inputs against discrete ranges, without consideration of uncertainty.

To use discrete logic:

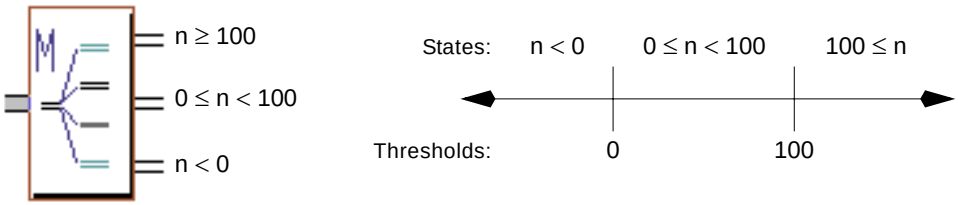
1. Specify Logic as discrete.
2. Specify numeric values for the Thresholds attributes:
   a. Click on the Threshold attribute whose value you want to edit. This puts your cursor in the dialog at the bottom of the configuration panel.
   b. Enter a new value in the dialog and press Return.

Note The configuration panel for the Multi-state block converts integers to floating point numbers.

The first threshold in the list corresponds with the top-most output port and should be the largest number. The numbers should decrease as you go down the list and down the output ports. GDA validates these values when you select Apply or OK.

The block tests whether an input value is above the largest threshold, below the smallest threshold, or between two thresholds.

For example, suppose you create a Multi-state block to classify temperature into three mutually exclusive states: below 0 degrees, between 0 and 100 degrees, and above 100 degrees. The associated threshold values are 0.0 and 100.0 degrees, as shown in the following figure. The ranges represent the values of *n* that produce an output of .true.



Using discrete logic, the output path associated with a given threshold passes . true when the input path is within a discrete numeric range. The following table shows discrete values for each output state for a three-state observation block with thresholds 0.0 and 100.0. Notice the behavior of the output paths when the input value is exactly equal to a threshold value.

If the input value is...	ip-out-1 **passes...**	ip-out-2 **passes...**	ip-out-3 **passes...**
less than 0.0	.true	.false	.false

If the input value is...	ip-out-1 passes...	ip-out-2 passes...	ip-out-3 passes...
greater than or equal to 0.0 but less than 100.0	.false	.true	.false
greater than or equal to 100.0	.false	.false	.true

Using Fuzzy Logic

You use fuzzy logic to test numeric inputs against numeric thresholds and pass scaled belief values based on threshold uncertainties. As the input value increases or decreases, the output belief values make a smooth transition from 0.0 to 1.0, as opposed to passing either 0.0 or 1.0 using discrete logic.

To use fuzzy logic:

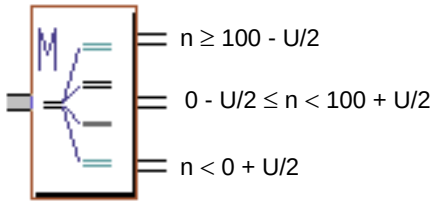
1. Specify Logic as fuzzy.
2. Specify numeric values for the Thresholds attribute. For details of how to edit attribute values in scroll areas, see "Using Discrete Logic" on page 260.
3. Specify numeric values for the Threshold Uncertainties attribute.

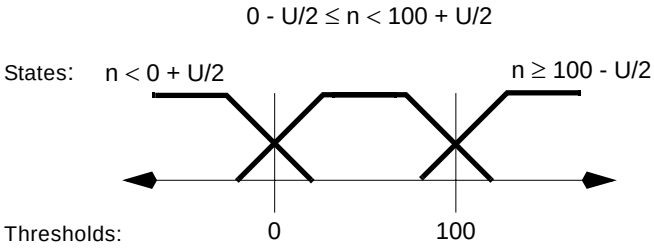
The first threshold in the list corresponds to the top-most output port and should be the largest number. The numbers should decrease as you go down the list and down the output ports. GDA validates these values when you select Apply or OK.

The first threshold uncertainty also corresponds to the top-most output port. You can specify different uncertainties for each threshold.

Note When Logic is fuzzy, Input Type is automatically numeric in the configuration panel; you cannot specify symbolic inputs using fuzzy logic. Also, when Logic is discrete, the Threshold Uncertainties attribute is grayed out.

The following figure illustrates a three-state observation block using fuzzy logic. The thresholds are 0.0 and 100.0. The numeric ranges specify U/2 as the uncertainty band around each threshold, where U is the Threshold Uncertainty. The number line shows the membership function for the three output states, and the associated ranges with the values of *n* that produce non-zero belief values.





Using fuzzy logic, the inference paths on either side of a given threshold pass scaled belief values when the input value is within an uncertainty band of the threshold. The scaled belief values are proportionally divided between the two output states, depending on where the input value lies with respect to the threshold.

If the input is between two thresholds and outside of the threshold uncertainties, the output state passes a belief value of 1.0 and a status value of .true, regardless of the threshold uncertainty.

If the input is exactly equal to one of the thresholds, the output state passes a Belief-value of 0.5 and a Status-value of unknown.

Note Normally, you specify thresholds and uncertainties so that a maximum of two output paths pass scaled belief values at one time. To do this, insure that the threshold plus half its threshold uncertainty is less than the next larger threshold minus half its threshold uncertainty.

The following table shows fuzzy belief values for a three-state observation block with thresholds of 0.0 and 100.0, and threshold uncertainties of 50.0 and 50.0. The table offsets the thresholds using shading.

If the input value is...	ip-out-1 **passes...**	ip-out-2 **passes...**	ip-out-3 **passes...**
less than or equal to -25.0	1.0	0.0	0.0
between -25.0 and 0.0	0.5 to 1.0	0.0 to 0.5	0.0
0.0	0.5	0.5	0.0

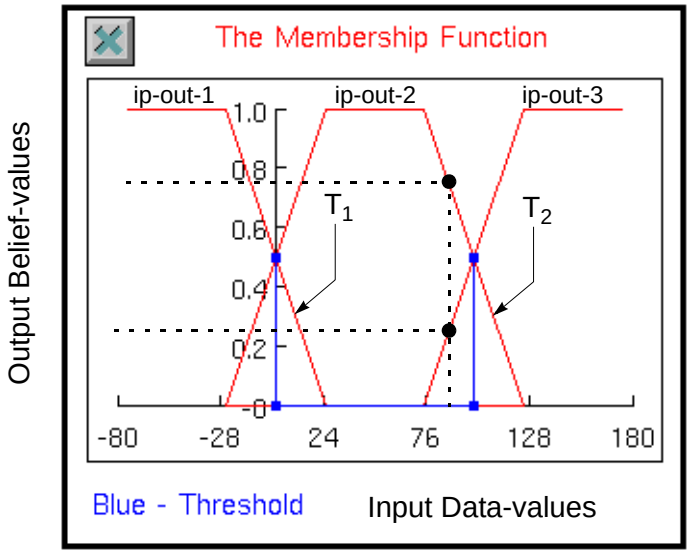
If the input value is...	ip-out-1 passes...	ip-out-2 passes...	ip-out-3 passes...
between 0.0 and 25.0	0.0 to 0.5	0.5 to 1.0	0.0
greater than or equal to 25.0 and less than or equal to 75.0	0.0	1.0	0.0
greater than 75.0 but less than 100.0	0.0	0.5 to 1.0	0.0 to 0.5
100.0	0.0	0.5	0.5
greater than 100.0 but less than 125.0	0.0	0.5 to 0.0	0.5 to 1.0
greater than or equal to 125.0	0.0	0.0	1.0

For a graphical representation of fuzzy belief values mapped against input values, see "Showing the Membership Function" below.

Showing the Membership Function

When Logic is fuzzy, show membership function appears as a choice on the block's menu. Selecting this menu choice displays a graph with output belief values along the vertical axis and input data values along the horizontal axis. Each observation appears as a separate function in red. Each threshold appears as a vertical line in blue. Observation states within a range appear as trapezoids, and observation states within an open-ended range appear as open-ended trapezoids.

The following figure shows the membership function for a three-state observation block with threshold values of 0 and 100, and threshold uncertainties of 50 and 50. The diagram labels the output paths and thresholds. The diagram also shows the output belief values corresponding to an input value of 87.5.



Specifying the Output Uncertainty

You specify Output Uncertainty for the Multi-state block to determine the Status-value of the output inference paths based on the Belief-value; in other words, you use Output Uncertainty to determine whether the output paths pass a value of .true, .false, or unknown. The Status-value depends on the output Belief-value; it does not depend on the input data value.

The following table summarizes the output Status-value based on the output Belief-value and the Output Uncertainty (OU).

The output Status-value is...	When the output Belief-value is...
.true	greater than 0.5 + OU/2
unknown	between 0.5 - OU/2 and 0.5 + OU/2
.false	less than 0.5 - OU/2

Using Symbolic and Textual Inputs

You use symbolic input types to test whether a symbolic or textual input is one of the specified symbolic categories. For example, you can create a Multi-state observation block associated with a pump whose input values indicate the status of the pump's operation, for example, running, warm standby, cold standby, and offline. Depending on the symbolic value of the discrete input, different downstream actions might occur.

The block tests whether a symbolic or textual input value is equal to one of the symbolic categories. If so, the output path passes .true; otherwise it passes .false.

To use symbolic inputs:

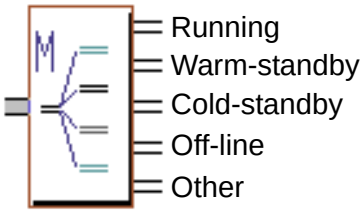
1. Specify Logic as discrete.
2. Specify Input Type as symbolic or textual.
3. Specify symbolic or textual values for the Categories attribute. For details of how to edit attribute values in scroll areas, see "Using Discrete Logic" on page 260.

The last Categories attribute in the list has a value of other, which passes .true if no other output path passes .true. The other Categories attributes contain the symbol user-default-symbol, by default.

Note The bottom Categories attribute always has the name Other; if you rename the attribute and select Apply, the name goes back to Other. Also, the Other category can only appear at the bottom of the list; if you enter Other in another position and select Apply, GDA inserts the default value.

Note For symbolic or textual inputs, specify the same number of categories as the number of output states. Compare this with numeric inputs where you specify one fewer thresholds than the number of states.

The following figure shows a Multi-state block with five output states and five symbolic threshold values. The output paths pass .true when the input value is in the specified category. If the input is not in any of the user-defined symbolic categories, the bottom-most output path passes .true.



Note If Input Type is symbolic or textual, the Logic attribute is automatically discrete in the configuration panel. Also, the Categories attribute is grayed out when Input Type is numeric.

Providing Explanations for Each Output State

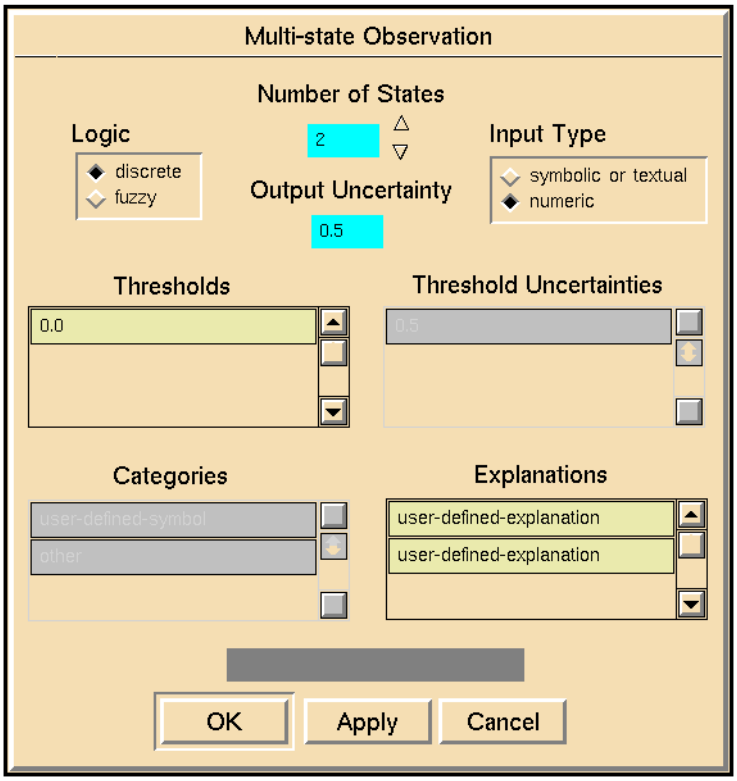
You can provide explanations to appear in the Explanation Queue based on the status value of the output path. You display the explanation using the current explanation menu choice. This menu choice is only available for certain blocks, for example, the blocks on the Condition palette in the Inference Blocks menu.

To provide explanations, specify text strings for the Explanations attribute. The first explanation in the list corresponds to the top-most output port. You specify the same number of explanations as the number of output states for the block. The Explanations attributes contain user-defined-description by default.

For detailed of how to enter attribute values in scroll areas on the configuration panel, see “Using Discrete Logic” on page 260.

Configuring

This is the configuration panel for the Multi-state block.



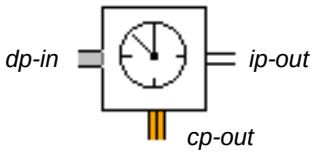
Attribute	Description
Number of States	The number of output paths that the block has, and thus the number of observation states.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Logic	Whether the block uses discrete or fuzzy logic when computing outputs based on numeric input values. See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Input Type	Whether the input values to the block are symbolic or textual, or numeric.
Threshold	The values against which the block compares its inputs to determine the output inference values.
Threshold Uncertainties	The uncertainty bands around the Threshold values that the block uses to determine in which output state numeric input values belong.
Categories	The symbolic or textual values that the block uses to determine in which output state symbolic or textual input values belong.
Explanations	Textual explanations associated with each output state, which the block uses to explain each output condition.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*

Data Expiration



The Data Expiration block triggers an action when the input value expires. The block takes a data path as input and passes a control signal and an inference value based on the input path quality. When the block receives a new value, GDA initiates an internal timer; if the block does not receive a new value first, the block outputs a control signal when the input value expires.

You use this block with the Expiration-time attribute, which appears in the table for data and inference paths. For information on this attribute, see "The Expiration-Time Attribute" on page 72 in the *GDA User's Guide*.

For example, you can connect a Queue Message block to the output control path of the Data Expiration block, which outputs a message to the Message Queue when its data input value expires.

The following table shows the possible output values for the Data Expiration block for different values of the input path Quality attribute:

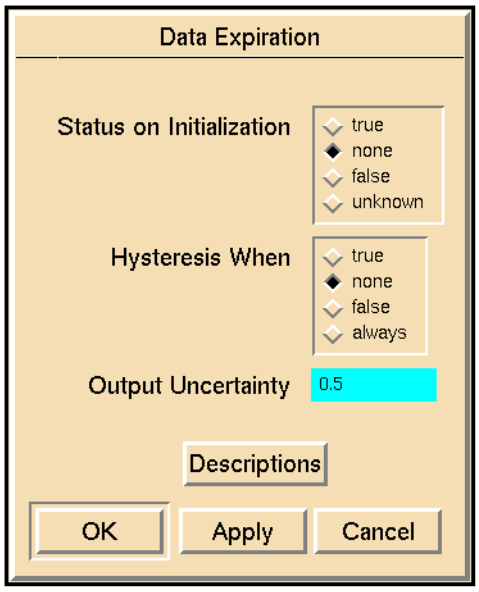
If input path Quality is...	Then the block passes...
ok	.false
manual	.false
expired	.true
no-value	unknown

Compare the Data Expiration block with the True When Expired block, which takes an inference path as input and determines whether a logical value has expired. See "True if Expired" on page 297 for more information.

You can use the Data Expiration block even if the global setting Propagate Expiration Events is no. See "The Expiration-Time Attribute" on page 72 in the *GDA User's Guide* for a description of this attribute.

Configuring

This is the configuration panel for the Data Expiration block.



Attribute	Description
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Hysteresis When	See “Specifying Hysteresis” on page 103 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*

10

Logic Gates

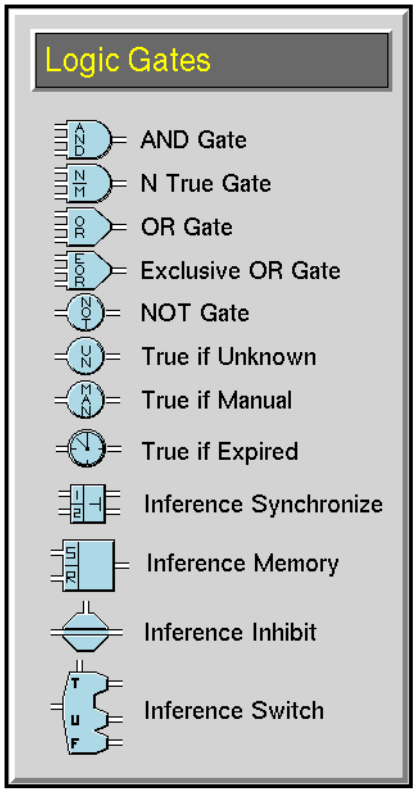
Describes the blocks the work with discrete and fuzzy inference values.

Introduction

Several blocks in GDA combine inference values and control the flow of inference data. Some blocks on the Logic Gates palette perform logical operations such as AND, OR, and NOT. These blocks work with discrete and fuzzy logic. Other

blocks control inference paths, such as pausing, inhibiting, and choosing among inference paths.

You can find the Logic Gates palette under the Inference submenu of the Palettes menu:



Performing Logical Operations

The logic gates perform basic logical operations on inference values. All these blocks have an attribute Logic which you can set to discrete or fuzzy. When Logic is discrete, the block operates on the status values .true, .false, and unknown. When Logic is fuzzy, the block operates on numeric belief values, which range from 0.0 to 1.0.

- The AND Gate block on page 275: When Logic is discrete, it passes .true if all its input status values are .true. When Logic is fuzzy, it passes the minimum of its belief values.
- The N True Gate block on page 279: When Logic is discrete, it passes .true if N or more of its input status values are .true. When Logic is fuzzy, it passes the Nth highest input belief value. N is an attribute you can set.

- The OR Gate block on page 283: When Logic is discrete, it passes .true if any of its input status values are .true. When Logic is fuzzy, it passes the maximum of its belief values.
- The Exclusive OR Gate block on page 287: When Logic is discrete, it passes .true if one and only one of its input status values is .true. When Logic is fuzzy, it passes a belief value based on its two highest input belief values.
- The NOT Gate block on page 291: When Logic is discrete, it passes the opposite of its input status value. When Logic is fuzzy, it passes 1 minus its input belief value.

Testing Inference Path Attributes

These blocks each test attributes of its input path:

- The True if Unknown block on page 293 passes .true if its input status value is unknown.
- The True if Manual block on page 296 passes .true if the Quality of its input inference path is manual.
- The True if Expired block on page 297 passes .true if its input status value is expired.

Stopping and Pausing Paths

These blocks let you stop the flow of inference values or pause them until some condition is met:

- The Inference Synchronize block on page 299 synchronizes two input inference paths by pausing one value until the other is received.
- The Inference Inhibit block on page 303 lets you enable or disable an inference path depending on another inference value.
- The Inference Memory block on page 301 remembers whether one of its input paths has been true.

Splitting Paths

The Inference Switch block on page 305 lets you send an inference value down one of three paths, depending on the value of another inference value.

See Also

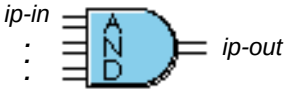
GDA has many other blocks that control the flow of information, but that take inference values or control signals as input:

- To combine belief values together, use the blocks in the Evidence Gates palette in the Inference Blocks menu. These blocks do not work with status values.
- To perform operations on a history of belief values, use some of the blocks on the Temporal Gates palette in the Inference Blocks menu.
- To control the flow of inference values depending on when they become .true or .false, use some of the blocks on the Tabular Gates palette in the Inference Blocks menu.

These blocks let you control the flow of data values or control signals:

- To control the flow of data values, use the blocks on the Data Control palette in the Data Blocks menu.
- To control the flow of control signals, use the blocks on the Control Actions palette in the Action Blocks menu.

AND Gate



When you are using discrete logic, the AND Gate passes .true if all its input status values are .true. When you are using fuzzy logic, it passes the minimum of all its input belief values.

Note The "Tabular AND Gates" on page 309 perform the same function as the AND Gate, but are useful when you have one input value that many AND Gates use.

Using Discrete Logic

This table lists what the AND Gate passes when Logic is discrete:

If…	It passes…
Any input is .false	.false
All inputs are .true	.true
Maximum Unknown Inputs or fewer are unknown, and the rest are .true	.true
More than Maximum Unknown Inputs are unknown, and the rest are .true	unknown
All inputs are unknown	unknown

Note If any input is .false, the AND Gate always passes .false, even if the number of unknown inputs is greater than Maximum Unknown Inputs.

How the Block Handles no-value Quality Inputs

An input path having a value of unknown and a Quality of no-value is not counted toward the number of Maximum Unknown Inputs.

For example, an AND Gate has 3 inputs and the Maximum Unknown Inputs is 1. If one of the inputs has a value of true, another a value of unknown and a Quality of OK, and the third a value of unknown and a Quality of no-value, the block

output is true. When determining whether the number of unknown inputs exceeds the Maximum Unknown Inputs, the third path is ignored.

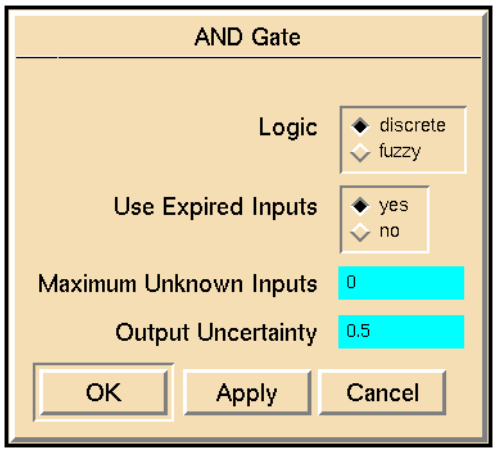
For more information about how blocks handle no-value inputs, see the *GDA User's Guide*.

Using Fuzzy Logic

When Logic is fuzzy, the block passes the minimum of its input values. In other words, it passes the input value that is closest to being completely false.

Configuring

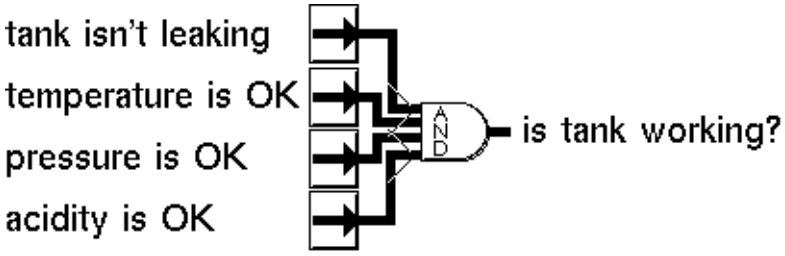
This is the configuration panel for the AND Gate.



Attribute	Description
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Use Expired Inputs	See "Determining Output Path Attributes for Peer Input Blocks" on page 76 in the *GDA User's Guide*.
Maximum Unknown Inputs	The number of inputs that can have a Status-value of unknown when determining whether the output inference value is .true or unknown.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

This figure shows an AND Gate connected to four Belief Entry Points:



This table shows some sample input and output values for the AND block in the previous figure, with Logic set to discrete and Maximum Unknown Inputs set to 2:

If the input values are…				The block passes…
.true	.false	.true	.true	.false
.true	.true	.true	.true	.true
unknown	unknown	.false	unknown	.false
unknown	.true	unknown	unknown	unknown
.true	.true	unknown	unknown	.true

This table shows some sample input and output values for an AND block, with Logic set to fuzzy:

If the input values are…				The block passes…
0.33	0.41	0.19	0.81	0.19
0.78	0.71	0.85	0.90	0.71

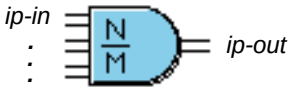
See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
Creating a Description for Logic Gates	page 99	*User's Guide*

For more information on...	See...	In this book...
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
The N True Gate	page 279	*Reference Manual*
The Tabular AND Gates	page 309	*Reference Manual*
The Weighted Evidence Combiner	page 327	*Reference Manual*
The Fuzzy Evidence Gate	page 332	*Reference Manual*
The Min Belief Gate	page 348	*Reference Manual*

N True Gate



When you are using discrete logic, the N True Gate passes .true if N or more of its input status values are .true. When you are using fuzzy logic, it passes the Nth highest input belief value. N is an attribute you can set.

This gate lets you implement voting. For example, if you have several redundant sensors, you can set up this gate to pass .true if a majority of the sensors are .true. This gate also lets you determine whether some minimal conditions are met. For example, you can use this gate to determine whether a sufficient number of sensors are working or whether sufficient resources are available.

Using Discrete Logic

This table lists what the N True Gate passes, when Logic is discrete. The attribute N is the target number of .true inputs.

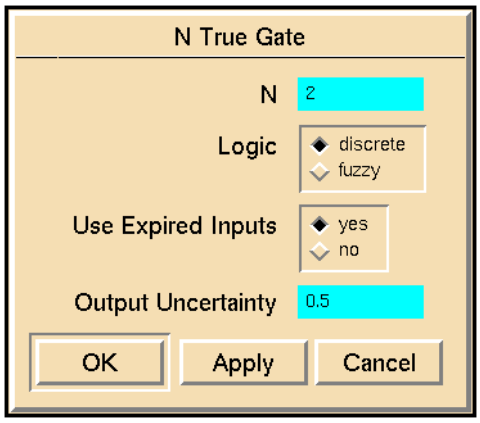
The block passes...	When...
.true	The number of .true inputs is greater than or equal to N.
unknown	The number of .true inputs is less than or equal to N, and the number of .true inputs plus the number of unknown inputs is greater than or equal to N.
.false	The number of .true inputs plus the number of unknown inputs is less than N.

Using Fuzzy Logic

When Logic is fuzzy, the block passes the Nth highest input belief value.

Configuring

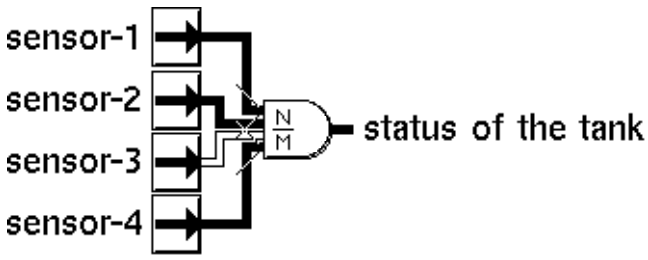
This is the configuration panel for the N True Gate.



Attribute	Description
N	The number of .true inputs that the block must have to pass a value of .true.
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Use Expired Inputs	See "Determining Output Path Attributes for Peer Input Blocks" on page 76 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

This figure shows an N True Gate connected to four Belief Entry Points. It determines whether at least 3 of the 4 sensors have the value .true:



This table shows some sample input and output values for the N True Gate in the previous figure, with Logic set to discrete and N set to 3:

If the input values are…				The block passes…
.false	.false	.true	.true	.false
.false	.true	.true	.true	.true
.true	.true	.true	.true	.true
unknown	unknown	.false	unknown	.false
unknown	.true	unknown	unknown	unknown
.true	.true	.true	unknown	.true

This table shows some sample input and output values for an N True block, with Logic set to fuzzy and N set to 3:

If the input values are…				The block passes…
0.33	0.41	0.19	0.81	0.33
0.78	0.71	0.85	0.90	0.78

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
Creating a Description for Logic Gates	page 99	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
The AND Gate	page 275	*Reference Manual*
The Tabular AND Gates	page 309	*Reference Manual*
The Weighted Evidence Combiner	page 327	*Reference Manual*

For more information on...	See...	In this book...
The Fuzzy Evidence Gate	page 332	*Reference Manual*
The Min Belief Gate	page 348	*Reference Manual*

OR Gate

When you are using discrete logic, the OR Gate passes .true if any of its input status values are .true. When you are using fuzzy logic, it passes the maximum of its belief values.

Note The "Tabular OR Gates" on page 314 perform the same function as the OR Gate, but are useful when you have one input value that many OR Gates use.

Using Discrete Logic

This table lists what the OR Gate passes, when Logic is discrete:

If...	It passes
Any input is .true	.true
All inputs are .false	.false
Maximum Unknown Inputs or fewer are unknown, and the rest are .false	.false
More than Maximum Unknown Inputs are unknown, and the rest are .false	unknown
All inputs are unknown	unknown

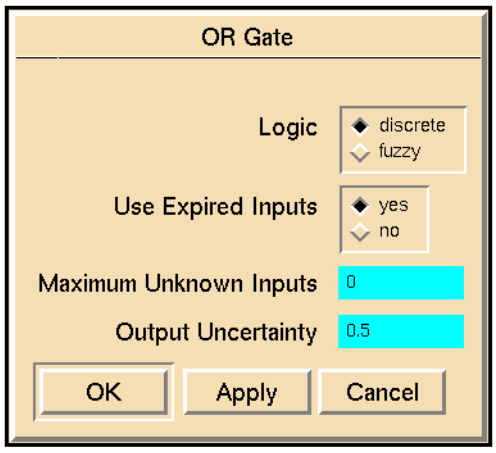
Note If any input is .true, the OR Gate always passes .true, even if the number of unknown inputs is greater than Maximum Unknown Inputs.

Using Fuzzy Logic

When Logic is fuzzy, the block passes the maximum of its input values. In other words, it passes the input value that is closest to being completely true.

Configuring

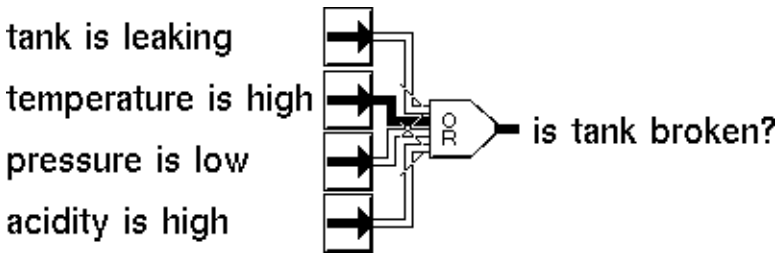
This is the configuration panel for the OR Gate.



Attribute	Description
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Use Expired Inputs	See "Determining Output Path Attributes for Peer Input Blocks" on page 76 in the *GDA User's Guide*.
Maximum Unknown Inputs	The number of inputs that can have a Status-value of unknown when determining whether the output inference value is .false or unknown.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

This figure shows an OR Gate connected to four entry points:



This table shows some sample input and output values for the OR block in the previous figure, with Logic set to discrete and Maximum Unknown Inputs set to 2:

If the input values are…				The block passes…
.true	.false	.true	.true	.true
.false	.false	.false	.false	.false
unknown	unknown	.false	unknown	unknown
unknown	.true	unknown	unknown	.true
.true	.true	unknown	unknown	.true

This table shows some sample input and output values for an OR block, with Logic set to fuzzy:

If the input values are…				The block passes…
0.33	0.41	0.19	0.81	0.81
0.78	0.71	0.85	0.90	0.90

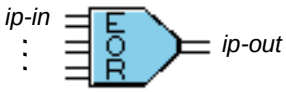
See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
Creating a Description for Logic Gates	page 99	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
The Exclusive OR Gate	page 287	*Reference Manual*
The Tabular OR Gates	page 314	*Reference Manual*
The Weighted Evidence Combiner	page 327	*Reference Manual*

For more information on...	See...	In this book...
The Fuzzy Evidence Gate	page 332	*Reference Manual*
The Max Belief Gate	page 346	*Reference Manual*

Exclusive OR Gate



When you are using discrete logic, the Exclusive OR Gate passes .true if one and only one of its input status values is .true. When you are using fuzzy logic, it passes a belief value based on its two highest input belief values.

Using Discrete Logic

This table lists what the Exclusive OR Gate passes, when Logic is discrete:

If…	It passes…
One input is .true, and Maximum Unknown Inputs or fewer are unknown	.true
More than one input is .true	.false
No input is .true, and Maximum Unknown Inputs or fewer are unknown	.false
One input or no inputs are .true and more than Maximum Unknown Inputs are unknown	unknown
All inputs are unknown	unknown

Note If more than one input is .true, the Exclusive OR Gate always passes .false, even if the number of unknown inputs is greater than Maximum Unknown Inputs.

How the Block Handles no-value Quality Inputs

An input path having a value of unknown and a Quality of no-value is not counted toward the number of Maximum Unknown Inputs.

For example, an Exclusive OR Gate has 3 inputs and the Maximum Unknown Inputs is 1. If one of the inputs has a value of true, another a value of unknown and a Quality of OK, and the third a value of unknown and a Quality of no-value, the block output is true. When determining whether the number of unknown inputs exceeds the Maximum Unknown Inputs, the third path is ignored.

For more information about how blocks handle no-value inputs, see the *GDA User's Guide*.

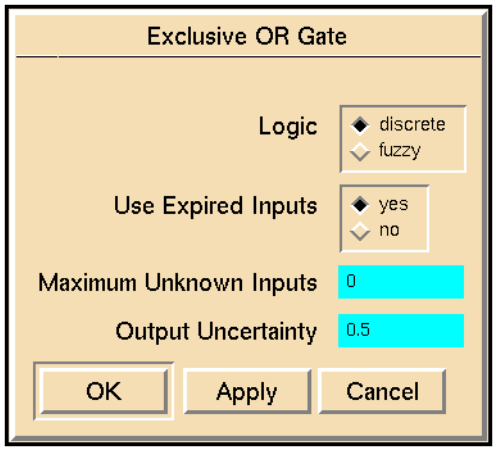
Using Fuzzy Logic

When Logic is fuzzy, the block computes its value with the two highest input values. It applies these rules, where the highest input value is I_1 and the second highest is I_2:

1. If $I_1 \leq 0.5$, it passes I_1.
2. If $I_2 \leq 0.5$, it passes the minimum of I_1 and $1.0\text{-}I_2$.
3. Otherwise, it passes $1.0\text{-}I_2$.

Configuring

This is the configuration panel for the Exclusive OR Gate.

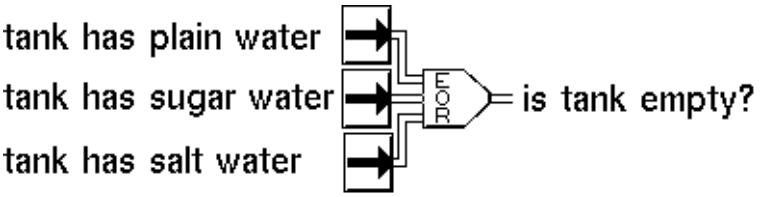


Attribute	Description
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Use Expired Inputs	See "Determining Output Path Attributes for Peer Input Blocks" on page 76 in the *GDA User's Guide*.

Attribute	Description
Maximum Unknown Inputs	The number of inputs that can have a Status-value of unknown when determining whether the output inference value is .false or unknown.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

This figure shows an Exclusive OR Gate connected to three entry points:



This table shows some sample input and output values for the Exclusive OR block in the previous figure, with Logic set to discrete and Maximum Unknown Inputs set to 1:

If the input values are…			The block passes…
.true	.false	.false	.true
.true	.true	.false	.false
.false	.false	.false	.false
unknown	.false	.true	.true
unknown	unknown	.false	unknown

This table shows some sample input and output values for an Exclusive OR block, with Logic set to fuzzy:

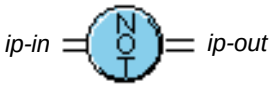
If the input values are…			The block passes…	Because …
0.17	0.38	0.25	0.38 (I_1)	I_1 (0.38) ≤ 0.5
0.41	0.19	0.81	0.59 (1.0–I_2)	I_1 (0.81) > 0.5, I_2 (0.41) ≤ 0.5, and min(1.0–I_2, I_1) = 1.0-I_2
0.71	0.85	0.90	0.15 (1.0–I_2)	I_1 (0.90) > 0.5 and I_2 (0.85) > 0.5

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Connecting to Peer Input Blocks	page 20	*User's Guide*
Creating a Description for Logic Gates	page 99	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
The OR Gate	page 283	*Reference Manual*
The Tabular OR Gates	page 314	*Reference Manual*
The Weighted Evidence Combiner	page 327	*Reference Manual*
The Fuzzy Evidence Gate	page 332	*Reference Manual*
The Max Belief Gate	page 346	*Reference Manual*

NOT Gate



When you are using discrete logic, the NOT Gate passes the logical opposite of its input status value. When you are using fuzzy logic, it passes 1 minus its input belief value.

Using Discrete Logic

When Logic is discrete, the NOT Gate passes the logical inverse of its input value, as this table shows:

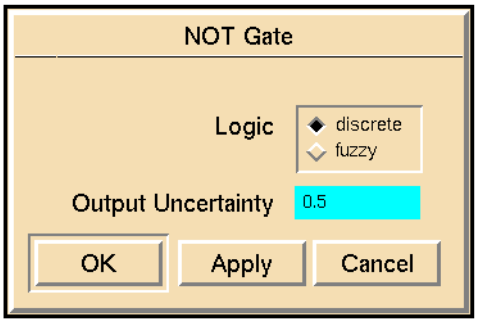
If the input is…	It passes…
.true	.false
.false	.true
unknown	unknown

Using Fuzzy Logic

When Logic is fuzzy, the block computes its output value by subtracting its input value from 1.0 (1.0 – Input).

Configuring

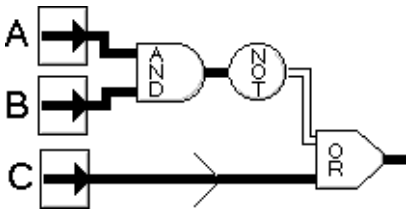
This is the configuration panel for the NOT Gate.



Attribute	Description
Logic	See “Specifying the Type of Logic to Use” on page 101 in the *GDA User’s Guide*.
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.

Example

This figure shows a diagram that implements NOT(A AND B) OR C:



See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Specifying the Type of Logic to Use	page 101	*User’s Guide*
Specifying Uncertainty	page 102	*User’s Guide*

True if Unknown

The True if Unknown gate determines if its input is unknown.

Using Discrete Logic

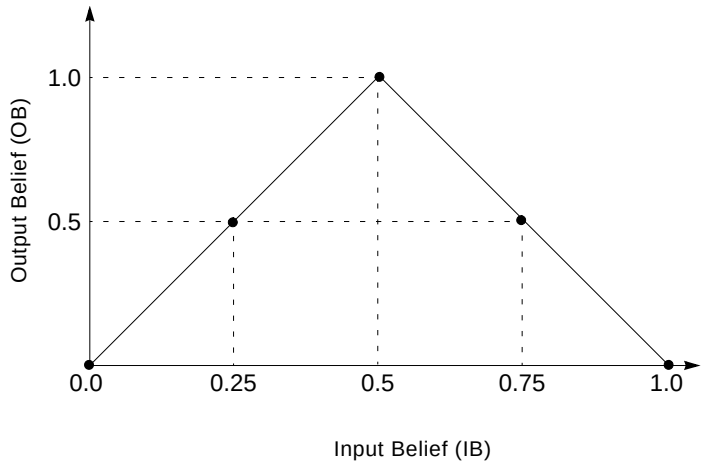
When Logic is discrete, this block passes .true if its input value is unknown. It passes .false, otherwise.

Using Fuzzy Logic

Using fuzzy logic, the block calculates the output belief value using the following formula, where OB is the output Belief-value and IB is the input Belief-value.

OB = (1.0 - (2.0 * abs (0.5 - IB)))

The following figure shows the membership function for this block:



The following table summarizes the output Status-value based on the output Belief-value (OB) and the Output Uncertainty (OU), which is an attribute you specify.

The output Status-value is...	When...
.true	OB ≥ 0.5 + OU/2

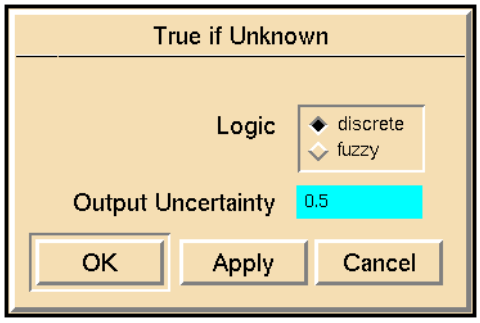
The output Status-value is...	When...
.false	OB ≤ 0.5 - OU/2
unknown	0.5 - OU/2 < OB < 0.5 + OU/2

The following table shows the output Belief-value and output Status-value for a set of input values, given an Output Uncertainty of 0.5 (the default):

When the input Belief-value is...	The output Belief-value is...	And the output Status-value is...
0.0	0.0	.false
0.25	0.5	unknown
0.5	1.0	.true
0.75	0.5	unknown
1.0	0.0	.false

Configuring

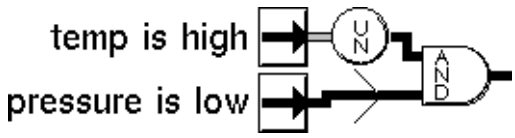
This is the configuration panel for the True if Unknown gate.



Attribute	Description
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

The AND Gate in the diagram in this figure passes .true if the temperature is unknown and the pressure is low:

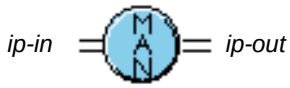


See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*

True if Manual



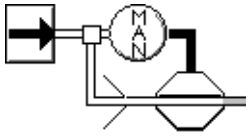
The True if Manual gate determines if the Quality of its input path is manual. The block passes .true if the Quality is manual. It passes .false otherwise.

Configuring

The True if Manual gate has no configurable attributes.

Example

In This figure, the True if Manual Gate inhibits a belief value from being passed if it was manually entered:



See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

True if Expired

The True if Expired gate determines when an inference input value expires. The block takes an inference path as input and passes an inference value based on the input path quality. The block initially passes a value of .false. When it receives a value, GDA initiates an internal timer; if the block does not receive a new value first, the block passes a value of .true when the value expires.

You use this block with the Expiration-time attribute, which appears in the table for inference paths. For information on this attribute, see "The Expiration-Time Attribute" on page 72 in the *GDA User's Guide*.

For example, you can connect a logical variable to the True if Expired block to test when the value expires.

The following table shows the possible output values for the True if Expired block for different values of the input Quality attribute:

If input Quality is...	Then the True if Expired block passes...
ok	.false
manual	.false
expired	.true
no-value	unknown

Compare the True if Expired block with the Data Expiration block, which takes a data path as input and outputs a control signal and an inference path. See "Data Expiration" on page 269 for more information.

You can use the True if Expired block even if the global setting Propagate Expiration Events is no. See "The Expiration-Time Attribute" on page 72 in the *GDA User's Guide* for a description of this attribute.

Configuring

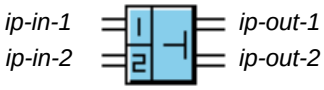
The True if Expired gate has no configurable attributes.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Inference Synchronize



The Inference Synchronize block synchronizes two inference paths by passing their values only after it has received values for both of them. After it has passed two values, it will pass a value again only after it receives two more values.

This block is useful when you must make sure that all the inference values in a flow diagram are up to date. For example, if two Entry Points send out values at slightly different times, the blocks connected to them will evaluate twice, once for each value. And the first time they evaluate they will be using one up-to-date value and one out-of-date value, so the results may be inaccurate. If the Entry Points are connected to a Inference Synchronize block, however, the values will go out simultaneously and the blocks connected to them will evaluate only once with two up-to-date values.

Resetting

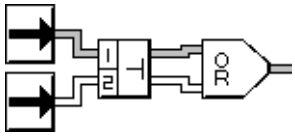
When you reset an Inference Synchronize block, it passes any pending input value.

Configuring

The Inference Synchronize block has no configurable attributes.

Example

This figure shows a flow diagram that uses a Inference Synchronize block:

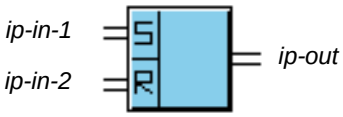


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Synchronize block	page 136	*Reference Manual*
The Event Sequence Gate	page 350	*Reference Manual*
The Control Synchronize block	page 461	*Reference Manual*

Inference Memory



The Inference Memory block remembers whether the top input (marked "S") has been true. The top input, marked "S," is the set path, and the bottom input, marked "R," is the reset path. When the set path becomes .true and the reset path is already .false, the block's value is "set" to .true. When the reset path becomes .true, the block's value is "reset" to .false.

When the block needs to account for unknown input values, it goes through the steps in this list in the order shown to determine what value to pass:

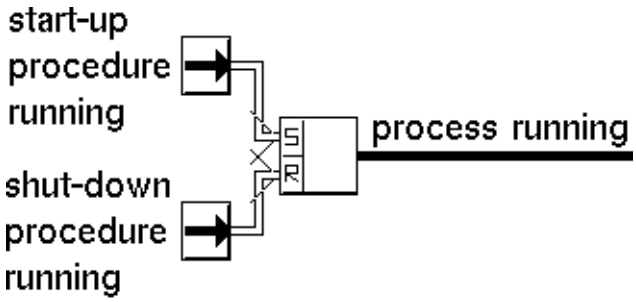
1. If R is .true, it passes .false.
2. If R is unknown, it passes unknown.
3. If S is .true, it passes .unknown.
4. If previous output is .true, it passes .true.
5. If S is unknown, it passes unknown.
6. If previous output is unknown on startup, it passes unknown.

Configuring

The Inference Memory block has no configurable attributes.

Example

The Inference Memory block in the following figure determines whether a process is running by remembering whether the process's start-up procedure has been run. In this diagram, the start-up procedure has just finished, so the process is running. The top Entry Point passes .true only while the start-up procedure is running, but not afterwards. The bottom Entry Point passes .true only while the shut-down procedure is running. The Inference Memory block remembers that the start-up procedure has run, even after the top Entry Point starts passes .false. And when the shut-down procedure runs, the block "forgets" that the start-up procedure was run and passes .false.

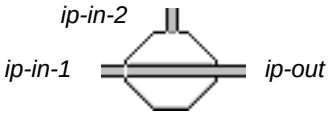


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Inference Inhibit



The Inference Inhibit block enables and disables an inference path. You can use it to control entire flow diagrams or an important subsection of one.

When the status value of the top inference path (ip-in-2) matches the value of the attribute Trigger On, the block inhibits the bottom input inference value (ip-in-1). When the block is inhibiting the inference path, it does not evaluate any attached chart or graph capabilities nor does it update their associated charts or graphs.

When the status value of the top inference path does not match Trigger On, the block passes the bottom input inference value normally.

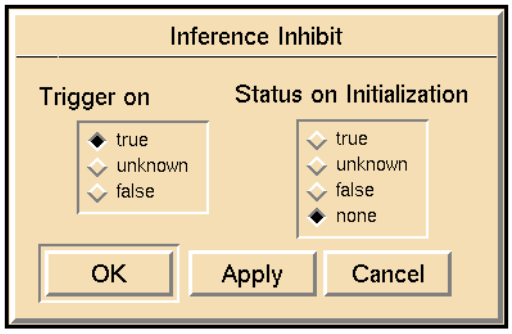
If Status on Initialization has a value, the block passes that value when it is initialized or reset; otherwise it passes nothing when initialized or reset.

Also, if the block has a Status on Initialization, and the top inference path value matches the Trigger On, the block passes the Status on Initialization.

GDA evaluates this block whenever either of the input ports receives a new value.

Configuring

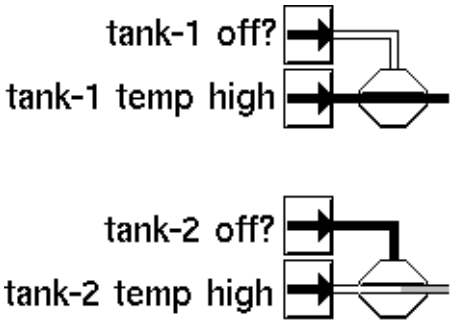
This is the configuration panel for the Inference Inhibit block.



Attribute	Description
Trigger On	The status value that determines when the block inhibits the flow of data.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.

Example

This figure shows a portion of a flow diagram that uses two Data Inhibit blocks to test whether a tank is on before analyzing its temperature. Trigger On is .true. Tank-1 is on and Tank-2 is off.

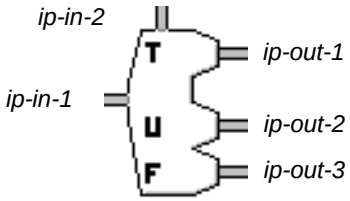


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Inhibit block	page 138	*Reference Manual*
The Control Inhibit block	page 447	*Reference Manual*

Inference Switch



The Inference Switch lets one inference value choose which path to send another inference value down. It is useful when you need to perform different diagnostics depending on some condition.

The Inference Switch passes values only when the inference value at port ip-in-1 changes; it does not pass values when the inference value at port ip-in-2 changes.

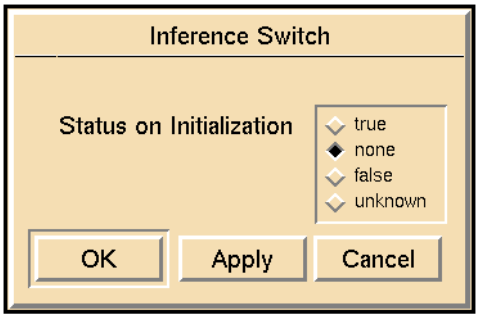
The value of the top input path (ip-in-2) determines where the value of left input path (ip-in-1) will go. This table shows which output path the value will go down:

If the top input is…	The left input is passed on the port labeled…
.true	“T” (ip-out-1)
unknown	“U” (ip-out-2)
.false	“F” (ip-out-3)

The attribute Status on Initialization determines the value of the two output paths that do not get the input value. Those paths pass the value of Status on Initialization.

Configuring

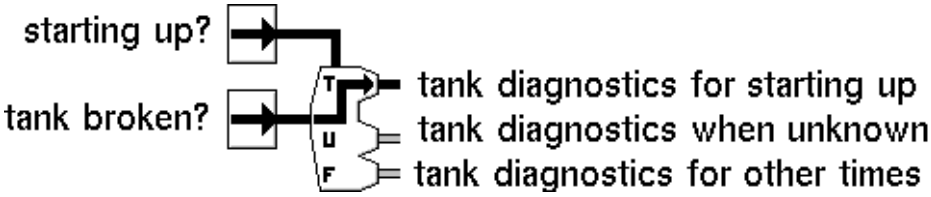
This is the configuration panel for the Inference Switch.



Attribute	Description
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.

Example

This figure shows a diagram that determines which tank diagnostics to run, depending on whether the process is starting up or not. Notice that an arrow on the Inference Switch icon shows you on which path the block is passing the value.



Since the top input path is .true, the output path labeled "T" passes the same value as the left input path. The other output paths pass unknown, the value of Status on Initialization.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Switch block	page 140	*Reference Manual*
The Control Switch block	page 454	*Reference Manual*

11

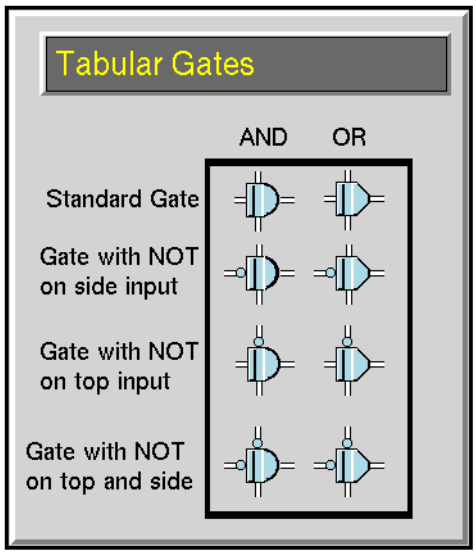
Tabular Gates

Describes the blocks that combine inference values, similar to using the AND, OR, and NOT Gates, but using a different block layout.

Introduction

The Tabular Gates perform the same function as the AND Gate and OR Gate, but are designed slightly differently. If using regular AND Gates and OR Gates forces you to cross paths in your diagram, these gates may help keep your diagram cleaner and simpler.

You can find the Tabular Gates palette under the Inference submenu of the Palettes menu:

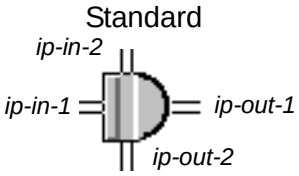


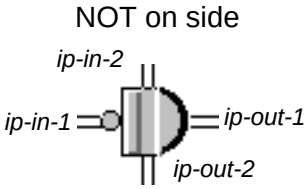
These are the differences between Tabular Gates and regular AND Gates and OR Gates:

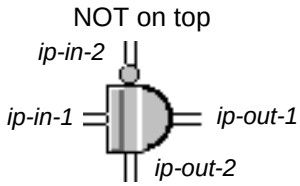
- Tabular Gates have only two input paths: one on the top and one on the left.
- Tabular Gates have an additional output path on the bottom, which passes the value of the top input path. This difference lets you create a diagram where one value is used in several different gates. Place Tabular Gates one on top of the other, and each gate passes its value from its top port to its bottom port.
- Some Tabular Gates have NOT indicators (small circles) on one or both input paths. The gate inverts the value of those inputs before performing its operation. If a NOT indicator is on the top input path, the gate also inverts that value before passing it down the bottom path.
- Tabular Gates do not have the attribute Maximum Unknown Inputs. They act as if the attribute were set to 0.

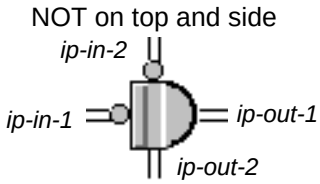
For more information on these gates and other logic gates, see “Logic Gates” on page 271.

Tabular AND Gates









The Tabular AND Gates perform a logical AND operation. They are useful when using an AND Gate would force you to cross paths and make a diagram difficult to understand, such as when one value is an input to more than one AND Gate.

The Standard Tabular AND Gate performs an AND on its two input paths, which are on the left and the top of the icon, and passes the result on the right output path. It also passes the value from its top input path to its bottom output path, without change.

The block passes…	On this port…
Top port AND left port	Right port (ip-out-1)
Top port	Bottom port (ip-out-2)

The other Tabular AND Gates have a NOT indicator (a small circle) on one or both input paths. These gates invert the input values that have a NOT indicator, perform an AND on the resulting values, and pass the result on the right output path. If the top port has a NOT indicator, the block passes the inverse of the top port on the bottom port.

The bottom port (ip-out-2) passes the top input path value (ip-in-2) or performs a NOT operation on the input, if there is a NOT indicator on top.

Using Discrete Logic

This table lists what the Standard Tabular AND Gate passes when Logic is discrete. Using a Standard Tabular AND Gate is like using an AND Gate with two inputs and Maximum Unknown Inputs set to 0.

If one input is…	and the other is…	The gate passes…
.true	.true	.true
.true	.false	.false
.false	.false	.false
unknown	.true	unknown
unknown	.false	.false
unknown	unknown	unknown

The other Tabular AND Gates have a NOT indicator (a small circle) on one or both input paths. If an input path has a NOT indicator on it, invert that input value and use the table above to determine the output value. Remember that the inverse of unknown is also unknown.

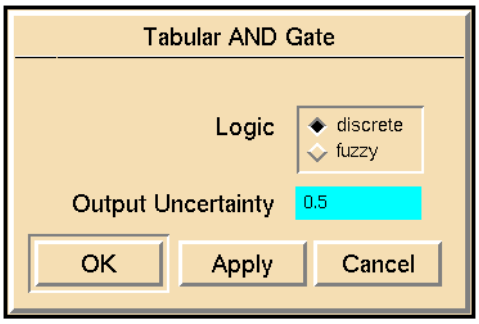
Using Fuzzy Logic

When Logic is fuzzy, the block passes the minimum of its input values. In other words, it passes the input value that is closest to being completely false.

The other Tabular AND Gates have a NOT indicator (a small circle) on one or both input paths. If an input path has a NOT indicator on it, invert that input value by subtracting it from 1.0 (1.0 - Input) and then compute the minimum of the two values.

Configuring

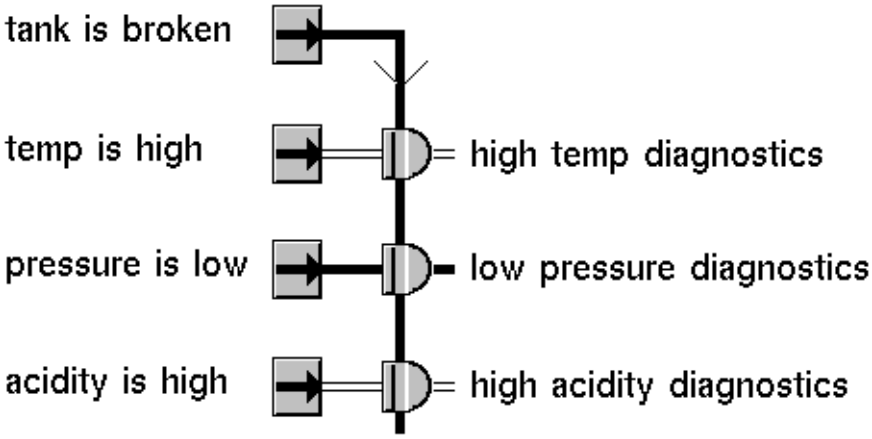
This is the configuration panel for the Standard Tabular AND Gate. The panels for the other AND gates are identical except for their names.



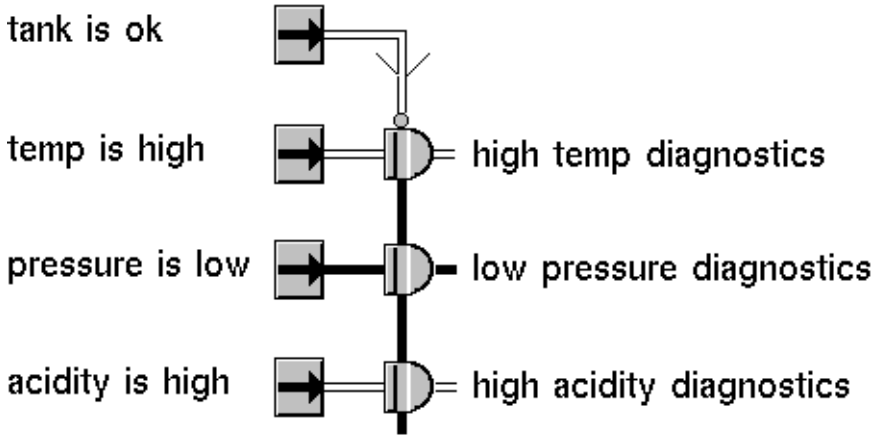
Attribute	Description
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

The following figure shows three Tabular AND Gates. Each is connected to the entry point labeled "tank is broken" and another entry point. Notice that if you created this diagram with regular AND Gates, you would have to let paths cross each other, making the diagram difficult to understand. This arrangement lets you run diagnostics when something suspicious happens only if the tank is also broken. For example, you do not want to run the high temp diagnostics if the tank is OK, even though the temperature is high.



This diagram works the same the previous diagram, even though the first Entry Point has the opposite meaning from the first entry point in the previous example. Notice that the first Tabular AND Gate has a NOT indicator on top. It inverts the Entry Point's value and passes the inverted value to the other Tabular AND Gates.



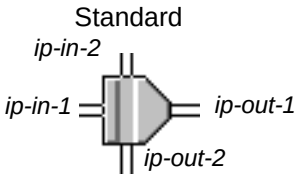
See Also

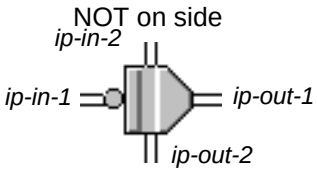
For more information on this block, see the sections below.

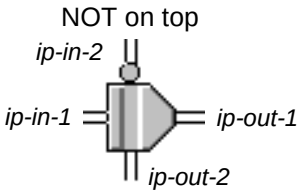
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
The AND Gate	page 275	*Reference Manual*

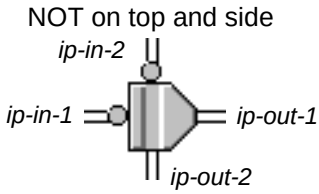
For more information on...	See...	In this book...
The NOT Gate	page 291	*Reference Manual*
Tabular OR Gates	page 314	*Reference Manual*
The Weighted Evidence Combiner Gate	page 327	*Reference Manual*
The Fuzzy Evidence Gate	page 332	*Reference Manual*

Tabular OR Gates









The Tabular OR Gates perform a logical OR operation. They are useful when using a regular OR Gate would force you to cross paths and make a diagram difficult to understand, such as when one value is an input to more than one OR Gate.

The Standard Tabular OR Gate performs an OR on its two input paths, which are on the left and the top of the icon, and passes the result on the right output path. It also passes the value from its top input path to its bottom output path, without change.

The block passes…	On this port…
Top port OR left port	Right port (ip-out-1)
Top port	Bottom port (ip-out-2)

The other Tabular OR Gates have a NOT indicator (a small circle) on one or both input paths. These gates invert the input values that have a NOT indicator, perform an OR on the resulting values, and pass the result on the right output path. If the top port has a NOT indicator, the block passes the inverse of the top port on the bottom port.

The bottom port (ip-out-2) passes the top input path value (ip-in-2) or performs a NOT operation on the input, if there is a NOT indicator on top.

Using Discrete Logic

The following table lists what the Standard Tabular OR Gate passes when Logic is discrete. Using a Standard Tabular OR Gate is like using an OR Gate with two inputs and Maximum Unknown Inputs set to 0.

If one input is…	and the other is…	The gate passes…
.true	.true	.true
.true	.false	.true
.false	.false	.false
unknown	.true	.true
unknown	.false	unknown
unknown	unknown	unknown

The other Tabular OR Gates have a NOT indicator (a small circle) on one or both input paths. If an input path has a NOT indicator on it, invert that input value and use the table above to determine the output value. Remember that the inverse of unknown is also unknown.

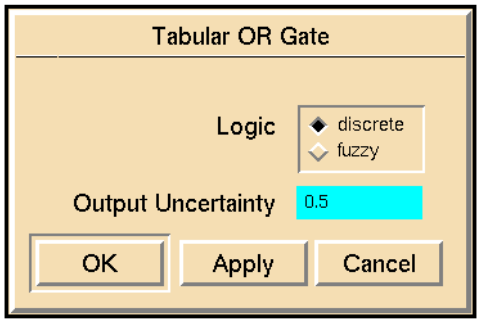
Using Fuzzy Logic

When Logic is fuzzy, the block passes the maximum of its input values. In other words, it passes the input value that is closest to being completely true.

The other Tabular OR Gates have a NOT indicator (a small circle) on one or both input paths. If an input path has a NOT indicator on it, invert that input value by subtracting it from 1.0 (1.0 - Input) and then compute the maximum of the two values.

Configuring

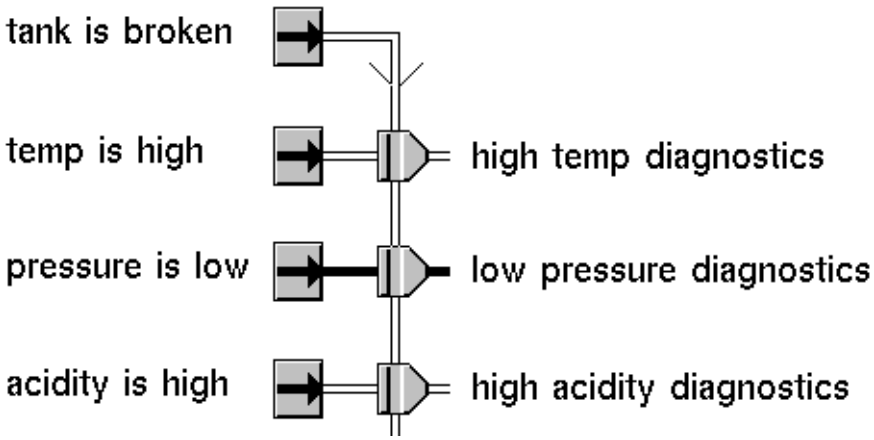
This is the configuration panel for the Standard Tabular OR Gate. The panels for the other OR gates are identical except for their names.



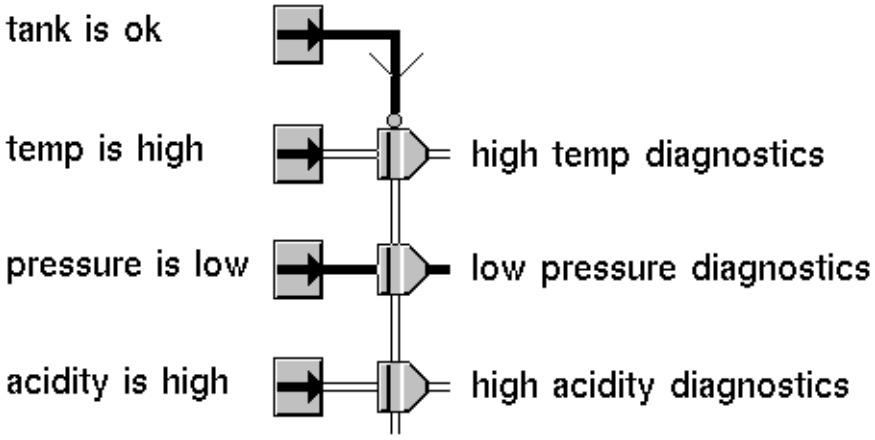
Attribute	Description
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

The following figure shows three Tabular OR Gates. Each is connected to the entry point labeled "tank is broken" and another entry point. Notice that if you created this diagram with regular OR Gates, you would have to let paths cross each other making the diagram difficult to understand. This arrangement lets you run some diagnostics when the tank is broken or when something suspicious happens (such as the tank's temperature is too high).



This diagram works the same as the previous diagram, even though the first Entry Point has the opposite meaning from the first entry point in the previous example. Notice that the first Tabular OR Gate has a NOT indicator on top. It inverts the Entry Point's value and passes the inverted value to the other Tabular OR Gates.



See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
The AND Gate	page 275	*Reference Manual*
The NOT Gate	page 291	*Reference Manual*
Tabular OR Gates	page 314	*Reference Manual*
The Weighted Evidence Combiner Gate	page 327	*Reference Manual*
The Fuzzy Evidence Gate	page 332	*Reference Manual*

12

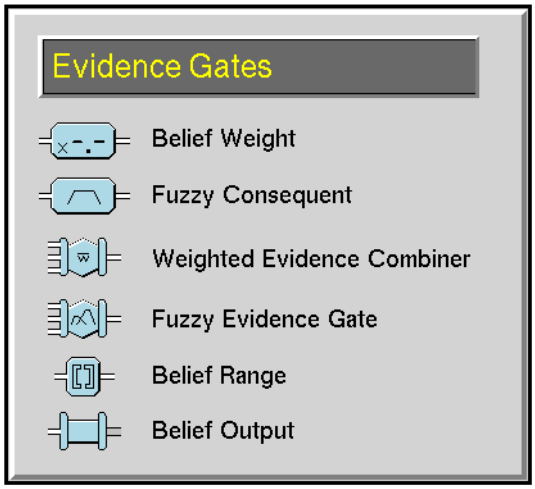
Evidence Gates

Describes the blocks that work exclusively with fuzzy logic values.

Introduction

The Evidence Gates are specifically designed to work with fuzzy logic values. They combine belief values together to draw conclusions, convert belief values to data values, and test whether a belief value falls into a certain range.

You can find the Evidence Gates palette under the Inference submenu of the Palettes menu:



Combining Belief Values

These blocks let you combine belief values together to draw new conclusions:

- The Weighted Evidence Combiner block on page 327 passes the weighted average of its input values. This block optionally applies a sigmoid function to the average to force the output value closer to 0.0 or 1.0. To specify a weight for an input value, attach a Belief Weight block to the input path.
- The Fuzzy Evidence Gate block on page 332 determines the certainty of an outcome given several rules. For example, the outcome might be "The tank will break," and the rules might be "If the tank is old, it's likely to break," "If the tank was cleaned recently, it's unlikely to break," and "If its temperature is high and its volume is low, it's almost certain to break." To specify the rules, attach the results of your inference blocks to Fuzzy Consequent blocks and attach those to the inputs of the Fuzzy Evidence Gate.

The Belief Weight and Fuzzy Consequent blocks are extensions to the Weighted Evidence Combiner and Fuzzy Evidence Gate. When attached to any other blocks, they have no affect. When attached to the correct block, they control how that block interprets the data from that input path.

Converting Belief Values

The Belief Output block on page 340 converts an input belief value to a data value.

Testing for Range

The Belief Range block on page 338 tests whether an input belief value is within a range you specify.

See Also

Here are some other blocks that perform similar functions:

- The Average Belief Gate, Max Belief Gate, and Min Belief Gate are also specifically designed to work with fuzzy logic values. They are on the Temporal Gates palette on the Inference Blocks menu.
- The blocks on the Logic Gates palette combine fuzzy logic values, and also work with discrete logic values.
- The Belief Input block on page 253 converts a data value to a belief value. It is on the Observations palette in the Inference Blocks menu.
- The In Range Value block or the Out of Range Value block on page 227 test whether data values are inside or outside a range. They are on the Observations palette in the Inference Blocks menu.

Belief Weight

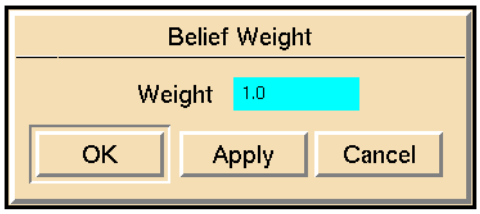
When connected to a Weighted Evidence Combiner, the Belief Weight block multiplies its input belief value by a fixed constant. When connected to any other block, the Belief Weight block passes its input value unchanged.

Specify the constant in the block's Weight attribute. The block displays the value of Weight on its icon.

Note Attach the Belief Weight block *only* to a Weighted Evidence Combiner. When connected to any other block, it has no affect. If you need to perform arithmetic on a belief value, convert the value to a data value with the Belief Output block on page 340, and use the Arithmetic blocks, described in "Arithmetic" on page 85.

Configuring

This is the configuration panel for the Belief Weight block.



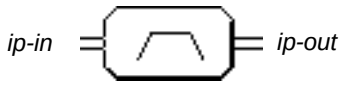
Attribute	Description
Weight	The constant value by which the block multiplies its input value.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Gain block	page 98	*Reference Manual*

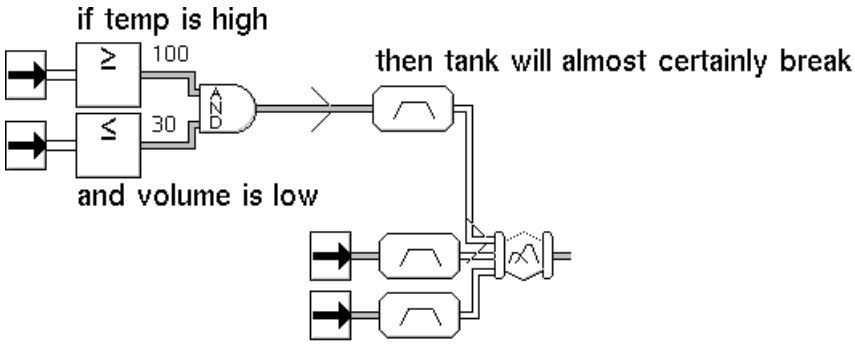
Fuzzy Consequent



You feed the output from a Fuzzy Consequent gate into a Fuzzy Evidence Gate, to describe the consequent of the fuzzy value, or a rule that determines how likely a conclusion is given a premise. The Fuzzy Consequent gate describes the likelihood that a conclusion is true, given a premise. You typically describe the premise by using Observations and Logic Gates connected to the Fuzzy Consequent's input path. The Fuzzy Evidence Gate combines the result of each Fuzzy Consequent together to determine how likely the conclusion is given all the rules.

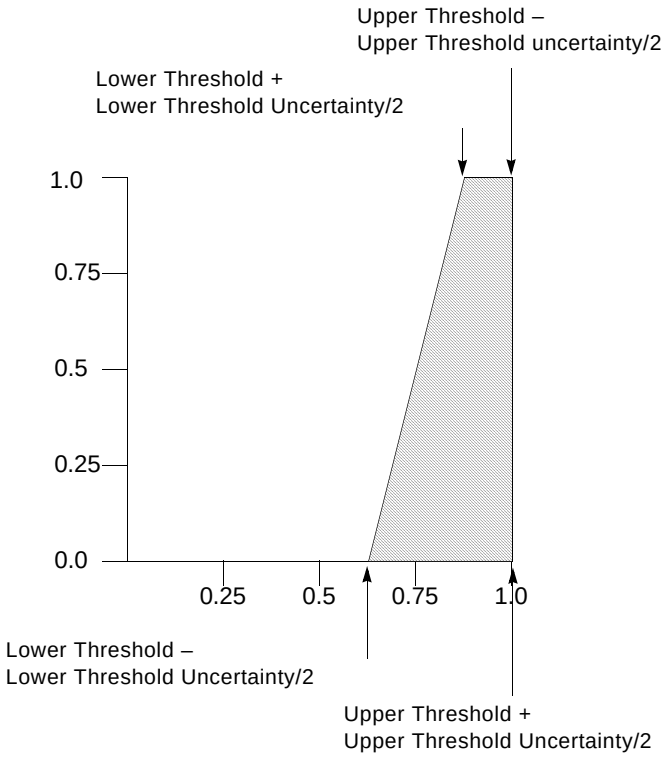
Note Attach the Fuzzy Consequent block *only* to a Fuzzy Evidence Gate. When the Fuzzy Consequent is connected to any other block, it has no affect.

For example, in the following figure, the two Observations and the top Fuzzy Consequent gate describe the rule "If the temp is high and the volume is low, then the tank is likely to break." The two Observations describe the rule's premise (the "If" part). The Fuzzy Consequent Gate describes the rule's consequent (the "Then" part). The other Fuzzy Consequent Gates might describe rules like "If the tank is old, it is likely to break," and "If the tank was cleaned recently, it is unlikely to break."



Specifying the Consequent

To specify the consequent – how likely the conclusion is given the premise – set the attributes Upper Threshold, Lower Threshold, Upper Threshold Uncertainty, and Lower Threshold Uncertainty. The block uses these attributes to create a membership function, like this:

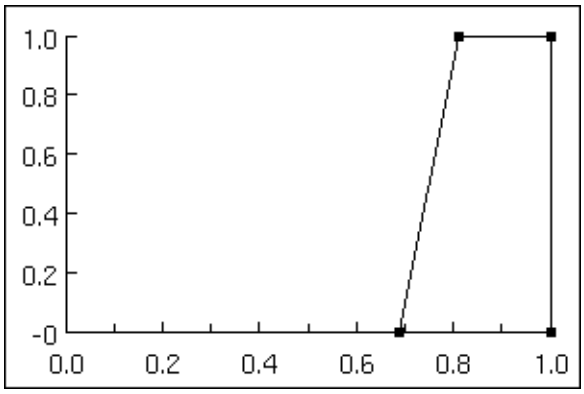


This table lists the attribute values for the previous figure:

Attribute	Value
Lower Threshold	0.75
Lower Threshold Uncertainty	0.25
Upper Threshold	1.0
Upper Threshold Uncertainty	0.0

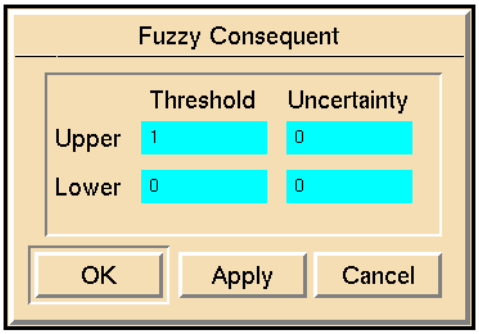
Viewing the Membership Function

To view a graph of a Fuzzy Consequent block's membership function, choose the menu choice show membership function from the block's menu. The following figure displays a graph that the show membership function menu choice created:



Configuring

This is the configuration panel for the Fuzzy Consequent block.



Attribute	Description
Upper Threshold	The upper end of the range in which the block tests its inputs to determine the output inference value. The Upper Threshold must be greater than the Lower Threshold.
Upper Threshold Uncertainty	An uncertainty band around the Upper Threshold value that the block uses to determine whether an input value is in range or out of range.

Attribute	Description
Lower Threshold	The lower end of the range in which the block tests its inputs to determine the output inference value. The Lower Threshold must be less than the Upper Threshold.
Lower Threshold Uncertainty	An uncertainty band around the Lower Threshold value that the block uses to determine whether an input value is in range or out of range.

See Also

For more information on this block, see the sections below.

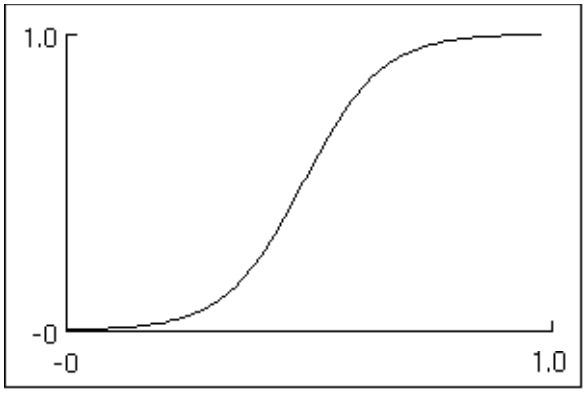
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Weighted Evidence Combiner

The Weighted Evidence Combiner computes a weighted average of its input belief values. The block passes the average on its output path. You specify weights by attaching Belief Weight blocks to the input paths.

You can also apply a sigmoid function to the weighted average, which forces values that are nearly true (or close to 1) to become very true (closer to 1) and values that are nearly false (close to 0) to become very false (closer to 0). For example, it could force a value in the high end of the uncertainty range to cross over the threshold and become true.

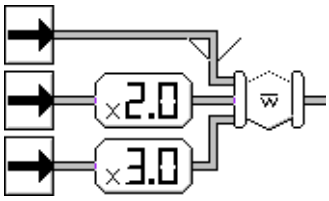
This figure shows a graph of a sigmoid function. The X-axis is the weighted average and the Y-axis is the result of the sigmoid function.



Specifying Weights

To specify the weight for a path, attach a Belief Weight gate to that input path and set the Belief Weight's Weight attribute. If an input path does not have a Belief Weight gate, this block uses a weight of 1.0. Notice that this block computes a simple average of its input values when there are no Weight Gates.

For example in this figure, the block computes its output value as if it were averaging six inputs: one with the top input value, two with the middle input value, and three with the bottom input value.



Specifying a Bias

To specify the constant that the block adds to the weighted average before passing it, set this block's Bias attribute. The block adds the Bias before applying the sigmoid function. If Bias is none or 0, this block does not add in a bias.

This block uses this equation to figure a weighted average, where Belief and Weight are input belief values and their weights, and Bias is an attribute of this block:

$$\text{Ave} = \frac{\sum \text{Belief} \times \text{Weight}}{\sum \text{Weight}} + \text{Bias}$$

Specifying a Sigmoid Function

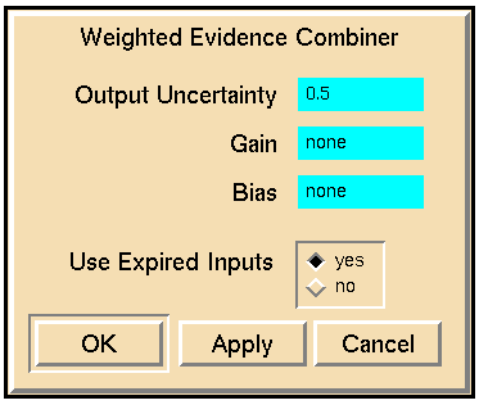
To apply a sigmoid function to the weighted average, set the block's Gain attribute to the slope that the function has as it passes through 0.5. If Gain is none, this block does not apply a sigmoid function and passes the weighted average unchanged.

This is the equation for the sigmoid function, where Gain is an attribute of this block and Ave is the weighted average that the equation above computes:

$$\frac{1}{1 + e^{(-\text{Gain} \times ((10 \times \text{Ave}) - 5))}}$$

Configuring

This is the configuration panel for the Weighted Evidence Combiner.



Attribute	Description
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Gain	The slope that the sigmoid function has as it passes through 0.5.
Bias	The constant that the block adds to the weighted average before passing it.
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

Examples

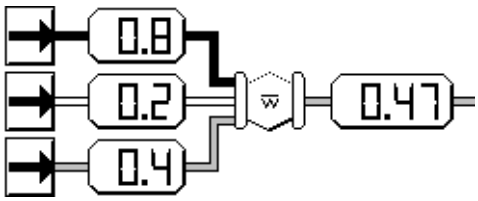
This section contains several examples of how a Weighted Evidence Combiner computes its output value. It shows how the block works with and without weights, with a Bias, and with a sigmoid function.

All these examples use the Belief Display blocks on page 505 to show what are the input values and output value of the block.

Computing an Unweighted Average

If the input belief values are not weighted, this block computes the average of its inputs, as shown in the following figure:

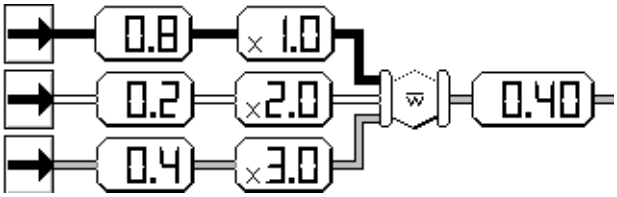
$$\left(\frac{0.8 + 0.2 + 0.4}{3} \approx 0.47\right)$$



Computing a Weighted Average

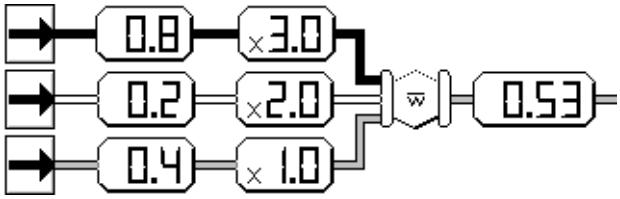
In this figure, the belief value 0.8 has a weight of 1.0, the belief value 0.2 has a weight of 2.0, and the belief value 0.4 has a weight of 4.0. The Weighted Evidence Combiner passes the value 0.4. It is as if the block had a total of six inputs: three of them being 0.4, two of them being 0.2, and one of them being 0.8. Notice that this block passes a lower value than the block in the previous figure, because the lower input values (0.2 and 0.4) are weighted more heavily than the highest input value (0.8):

$$\left(\frac{(0.8 \times 1.0) + (0.2 \times 2.0) + (0.4 \times 3.0)}{3.0 + 2.0 + 1.0} = \frac{0.8 + 0.4 + 1.2}{6.0} = 0.40\right)$$



Notice that if you switch the weights for two of the input values to weigh 0.8 more heavily and 0.4 less heavily, you get a much different result, 0.53, as this figure shows:

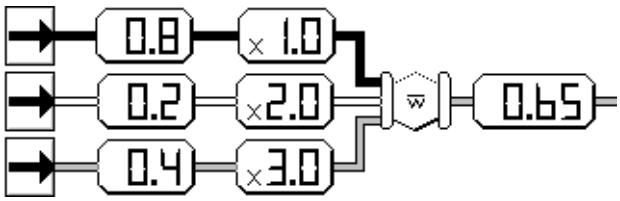
$$\left(\frac{(0.8 \times 3.0) + (0.2 \times 2.0) + (0.4 \times 1.0)}{3.0 + 2.0 + 1.0} = \frac{2.4 + 0.4 + 0.4}{6.0} \approx 0.53\right)$$



Computing a Biased Average

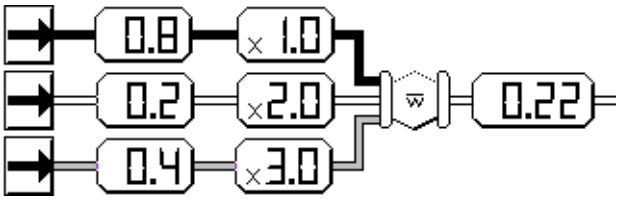
The Weighted Evidence Combiner in this figure uses the same input values and weights as in the example under "Computing a Weighted Average" on page 330, but it has a Bias of 0.25 and passes the value 0.65.

$$\left(\frac{(0.8 \times 1.0) + (0.2 \times 2.0) + (0.4 \times 3.0)}{3.0 + 2.0 + 1.0} + 0.25 = 0.40 + 0.25 = 0.65\right)$$



Computing a Weighted Average with a Sigmoid Function

The Weighted Evidence Combiner in this figure uses the same input values and weights as in the example under "Computing a Weighted Average" on page 330, but it has a Gain of 1.25 and applies a sigmoid function to its weighted average before passing it along. Notice that applying the sigmoid function forces the status value of this block to change from unknown to .false.



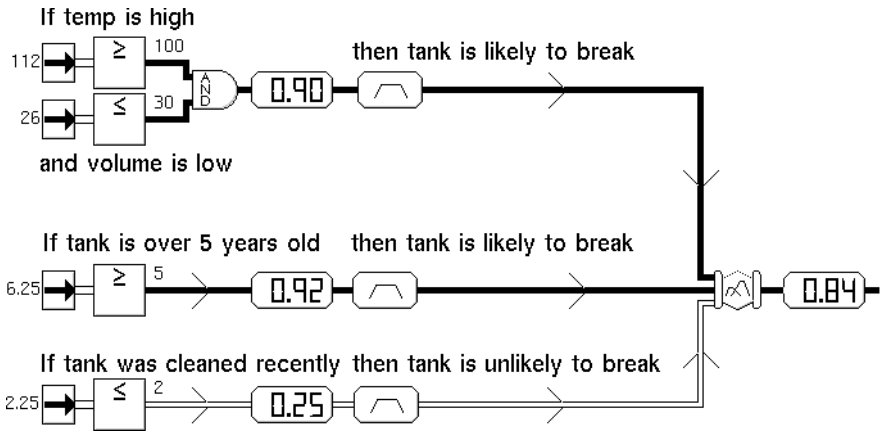
See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The AND Gate	page 275	*Reference Manual*
The OR Gate	page 283	*Reference Manual*
The N True Gate	page 279	*Reference Manual*

Fuzzy Evidence Gate

The Fuzzy Evidence Gate combines several rules to determine the possibility of a particular outcome. You specify the rules with Observations blocks and Fuzzy Consequent gates, as this figure shows:



Note All of a Fuzzy Evidence Gate's input paths must be connected to Fuzzy Consequent blocks. It ignores the value of any input path which is not connected to a Fuzzy Consequent block.

For example, the Fuzzy Evidence Gate in the figure above combines these three rules to determine how likely a tank is to break:

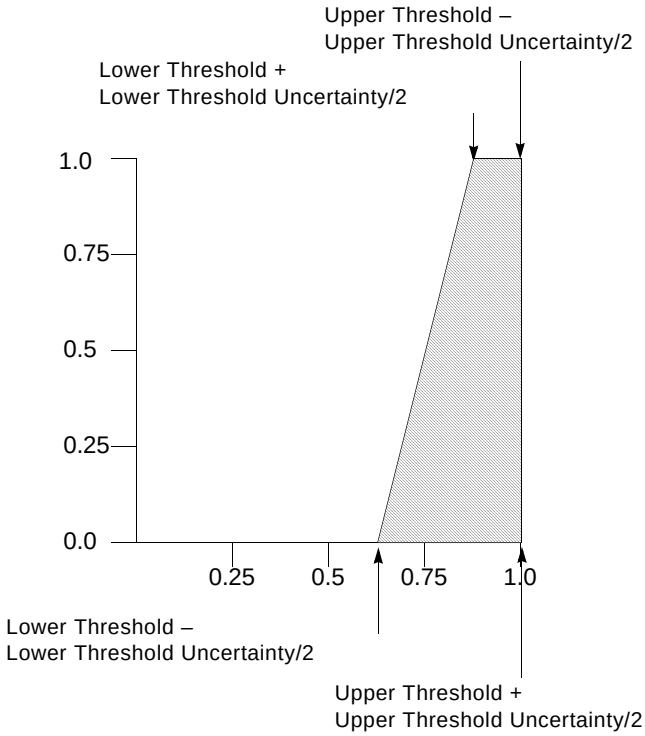
- If the temperature is high and the volume is low, then the tank is likely to break.
- If the tank is old, then it is likely to break.
- If the tank was cleaned recently, then it is unlikely to break.

In each rule, the Observations describe the rule's premise (the "If" part) and the Fuzzy Consequent Gate describes the rule's consequent (the "Then" part) or how likely the outcome is given the premise.

Specifying the Consequents

To specify the consequent – how possible the conclusion is given the premise – set the attributes Upper Threshold, Lower Threshold, Upper Threshold Uncertainty, and Lower Threshold Uncertainty in each Fuzzy Consequent block. The block

uses these attributes to create a function, like the one in this figure. This function is called the block's membership function.



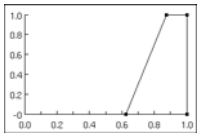
This table lists the attribute values for the figure above:

Attribute	Value
Lower Threshold	0.75
Lower Threshold Uncertainty	0.25
Upper Threshold	1.0
Upper Threshold Uncertainty	0.0

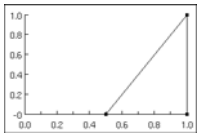
Combining the Consequents

When the Fuzzy Consequent receives an input belief value from its premise, the Fuzzy Evidence Gate computes the area of the polygon that is bounded by the Fuzzy Consequent's function and the input value. The Fuzzy Evidence Gate then combines all the polygons from its Fuzzy Consequents to form one shape and computes the center of gravity for that shape.

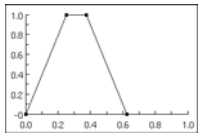
For example, the following figure shows how the example for the Fuzzy Evidence Gate block on page 332 determines its output value. The top three graphs are the membership functions for the Fuzzy Consequent blocks. The bottom graph shows the membership functions together. The areas bounded by the Fuzzy Consequent blocks' membership functions and input values are shaded. The areas in the membership functions for two Fuzzy Consequent blocks are darkly shaded, so the Fuzzy Evidence Gate counts them twice. The center of gravity for these three shapes, 0.84, is pointed out.



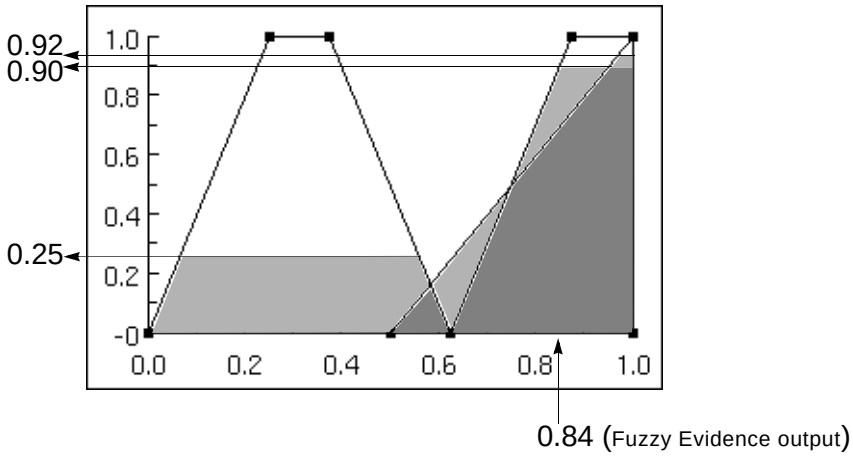
If temp is high and volume is low, then tank is likely to break



If tank is over 5 years old, then tank is likely to break



If the tank was cleaned recently, then tank is unlikely to break



Specifying Which Input Values to Ignore

If the belief value of an input path is less than the attribute Implication Threshold, the Fuzzy Evidence Block does not use that path's rule to compute its output value. For example, if the Implication Threshold of the Fuzzy Evidence Gate in the following figure were 0.30, the gate would ignore the input from the bottom input path (0.25) and the output would be 0.95.

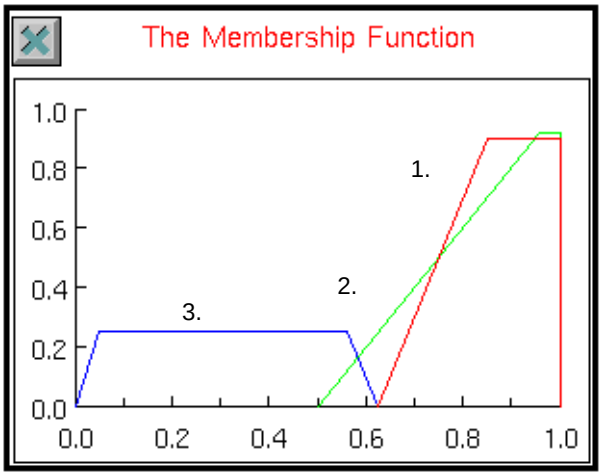
Showing the Membership Function

To display the membership function for the Fuzzy Evidence Gate, use the show membership function menu option. This menu choice is available when you connect the Fuzzy Consequent block to the inputs of the Fuzzy Evidence Gate. The function displays the membership function for each Fuzzy Consequent used as input to the Fuzzy Evidence Gate. The vertical extent of the membership

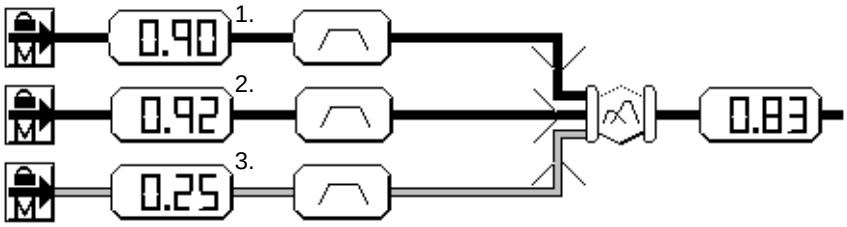
function for each Fuzzy Consequent input is the belief value passed by the block's inference path.

The function uses one of 8 colors for each input. If the block has more than 8 inputs, GDA recycles the colors.

The following figure shows the membership function for the example for the Fuzzy Evidence Gate block on page 332. The labeled components of the membership function correspond with the following example.



The labels in this example correspond to the membership function above:



The following table shows the attribute values for each Fuzzy Consequent Gate in the previous example:

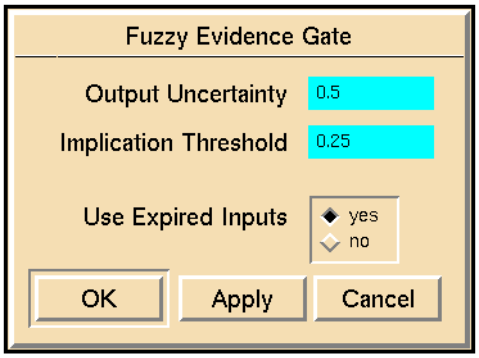
The Fuzzy Consequent labeled...	Has attributes...	Whose values are...
1	Upper Threshold	1.0
	Lower Threshold	0.75
	Upper Threshold Uncertainty	0.0
	Lower Threshold Uncertainty	0.25
2	Upper Threshold	1.0
	Lower Threshold	0.75
	Upper Threshold Uncertainty	0.0
	Lower Threshold Uncertainty	0.5
3	Upper Threshold	0.5
	Lower Threshold	0.1
	Upper Threshold Uncertainty	0.25
	Lower Threshold Uncertainty	0.2

The following table shows the attribute values for the Fuzzy Evidence Gate in the example:

The attribute...	Has a value of...
Output Uncertainty	0.5
Implication Threshold	0.25

Configuring

This is the configuration panel for the Fuzzy Evidence Gate.



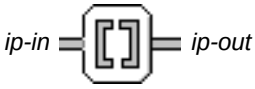
Attribute	Description
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Implication Threshold	The threshold for including an input belief value in the block's output computation. The block includes only those input belief values that are above the threshold.
Use Expired Inputs	See "Determining Whether a Block Uses Expired Inputs" on page 76 in the *GDA User's Guide*.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The AND Gate	page 275	*Reference Manual*
The OR Gate	page 283	*Reference Manual*
The N True Gate	page 279	*Reference Manual*

Belief Range

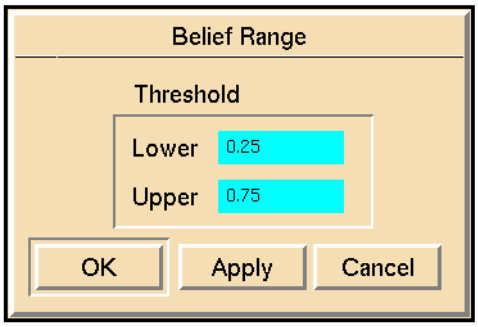


The Belief Range block detects whether the input belief value is within the range you specified with the attributes Lower Threshold and Upper Threshold. If the input value is within that range, the block passes the status value .true and the belief value 1.0. Otherwise, the block passes the status value .false and the belief value 0.0.

The brackets on the icon reflect the input and output status values. The left bracket takes on the color for the input status value. The right bracket takes on the color of the output status value.

Configuring

This is the configuration panel for the Belief Range block.

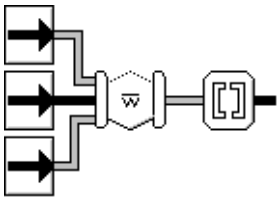


Attribute	Description
Lower Threshold	The lower end of the range in which the block tests its inputs to determine the output inference value. The Lower Threshold should be less than the Upper Threshold.
Upper Threshold	The upper end of the range in which the block tests its inputs to determine the output inference value.

Example

In this example, a Belief Range gate is testing whether the output of a Weighted Evidence Combiner is between 0.25 and 0.75. Since the Weighted Evidence

Combiner is passing 0.64, the Belief Range gate is passing the status value .true and the belief value 1.0.

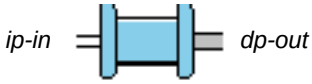


See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The In Range Value, Out of Range Value blocks	page 227	*Reference Manual*

Belief Output



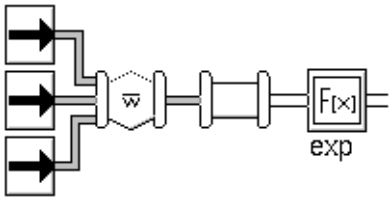
The Belief Output block converts its input belief value to a data value.

Configuring

The Belief Output block has no configurable attributes.

Example

In this example, a Belief Output gate converts the output of a Weighted Evidence Combiner to a data value, and an Arithmetic Function block takes the exponent of that value.



See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Belief Input block	page 253	*Reference Manual*

13

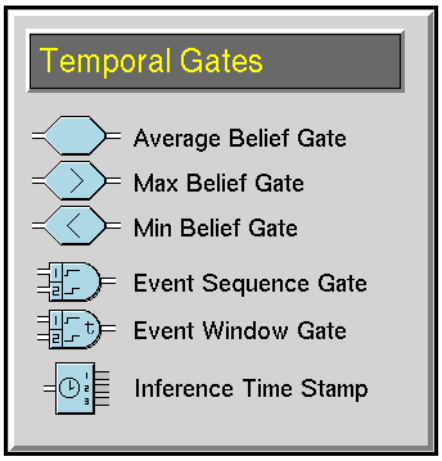
Temporal Gates

Describes the blocks that perform time-based reasoning.

Introduction

The blocks in the Temporal Gates palette let you perform temporal logic, such as operating on a history of inference values, passing on time stamps, and analyzing the timing of events.

You can find the Temporal Gates palette under the Inference submenu of the Palettes menu:



Performing Operations on a History

These gates let you perform an operation on a history of inference values. They are closely related to the blocks on the Time Series palette, which perform operations on a history of data values.

- The Average Belief Gate block on page 344 computes the average of the history of inference values.
- The Max Belief Gate block on page 346 computes the maximum of the history of inference values.
- The Min Belief Gate block on page 348 computes the minimum of the history of inference values.

Analyzing the Timing of Events

These blocks let you analyze when two different paths receive a status value:

- The Event Sequence Gate block on page 350 determines whether its top path receives a specified status value before its bottom path does. It passes .true only if the paths receive the correct value in the correct order.
- The Event Window Gate block on page 353 determines whether its top path receives a status value of true within a certain time window of the bottom path turning true.

Passing Time Stamps

The Inference Time Stamp block on page 359 has three output paths and passes the last time that its input inference path had a value of .true, .false, and unknown.

See Also

These are related blocks:

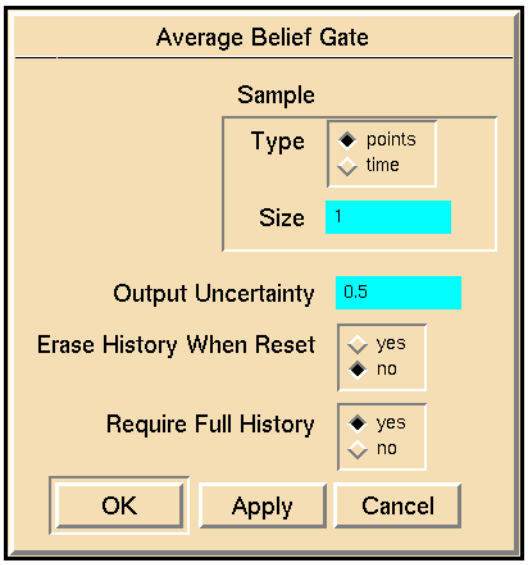
- The blocks on the Time Series palette operate on a history of data values. The Moving Average block computes the average of the history. The Moving Range block computes the maximum, minimum, and range (maximum – minimum) of the history.
- Some blocks on the Logic Gates palette in the Inference Blocks menu return the maximum and minimum of their input belief values. The AND Gate returns the minimum, and the OR Gate returns the maximum.
- The Data Time Stamp block on page 152 on the Data Control palette passes on the time it received a data value.
- Some blocks on the Logic Gates palette in the Inference Blocks menu let you control the timing of events, like Inference Synchronize and Inference Memory.

Average Belief Gate

The Average Belief Gate passes the average of its history of belief values.

Configuring

This is the configuration panel for the Average Belief Gate.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.

Example

If the block's history contains 0.9, 0.0, 0.5, 0.8, 0.2, and 0.6, it passes 0.5 or:

$$\frac{0.9 + 0.0 + 0.5 + 0.8 + 0.2 + 0.6}{6}$$

See Also

For more information on how to use this block, see the sections below.

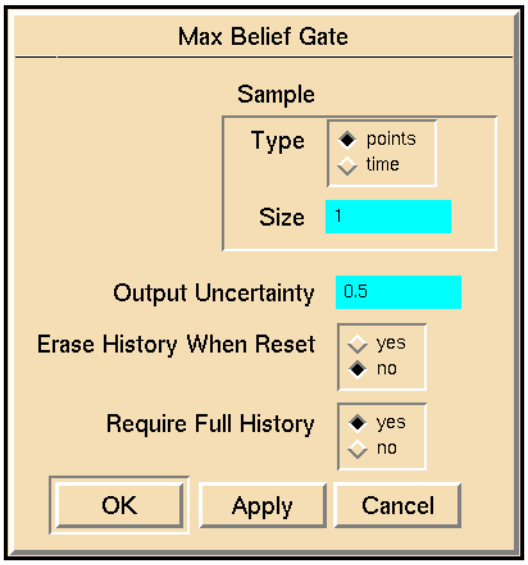
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
The Average Input Value block	page 104	*Reference Manual*
The Moving Range block	page 167	*Reference Manual*

Max Belief Gate

The Max Belief Gate passes the maximum of its history of belief values.

Configuring

This is the configuration panel for the Max Belief Gate.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.

Example

If the block's history contains 0.9, 0.0, 0.5, 0.8, 0.2, and 0.6, it passes 0.9, the maximum value.

See Also

For more information on how to use this block, see the sections below.

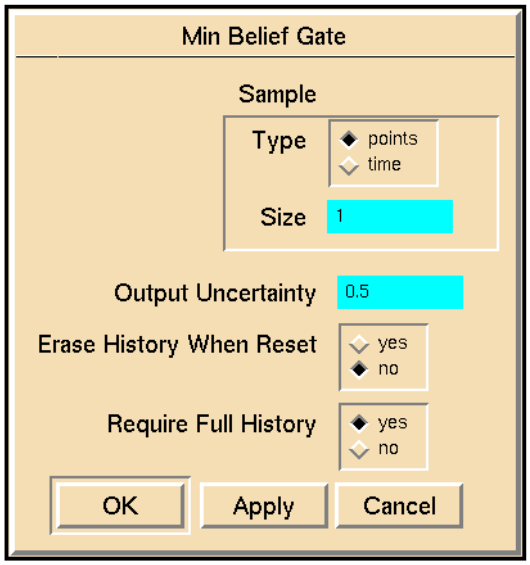
For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
The Low Limiting, High Limiting blocks	page 110	*Reference Manual*
The Low Selecting, High Selecting blocks	page 108	*Reference Manual*
The Moving Range block	page 167	*Reference Manual*
The OR Gate	page 283	*Reference Manual*

Min Belief Gate

The Min Belief Gate passes the minimum of its history of belief values.

Configuring

This is the configuration panel for the Min Belief Gate.



Attribute	Description
Sample Type and Sample Size	See "Specifying the Size of the History" on page 85 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Erase History when Reset	See "Specifying What Happens to History Upon Reset" on page 90 in the *GDA User's Guide*.
Require Full History	See "Specifying What to Do With Partial History" on page 91 in the *GDA User's Guide*.

Example

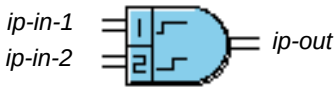
If the block's history contains 0.9, 0.0, 0.5, 0.8, 0.2, and 0.6, it passes 0.0, the minimum value.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Maintaining a History of Values	page 84	*User's Guide*
The Low Limiting, High Limiting blocks	page 110	*Reference Manual*
The Low Selecting, High Selecting blocks	page 108	*Reference Manual*
The Moving Range block	page 167	*Reference Manual*
The OR Gate	page 283	*Reference Manual*

Event Sequence Gate



The Event Sequence Gate detects when two input paths receive the value Trigger On in the proper order, that is the top input receives it before the bottom input. The attribute Trigger On must be either .true or .false.

This block is useful when two events must happen in a specific order. For example, you may need to ensure that a tank's agitator is on before its heating coils are on.

Specifying a Delay

To specify a lag time, set the attribute Temporal Uncertainty. The block must receive the value Trigger On from the bottom path at least Temporal Uncertainty seconds after it receives the value from the top path.

To specify what happens when the block receives the value Trigger On from the bottom path *exactly* Temporal Uncertainty seconds after it receives the value from the bottom path, set the attribute Enforce Strict Sequence. If Enforce Strict Sequence is yes, the block passes .false. If Enforce Strict Sequence is no, the block passes .true. Note that when Temporal Uncertainty is 0 seconds, Enforce Strict Sequence controls what happens when the two paths receive the value Trigger On simultaneously.

Using Discrete Logic

When Logic is discrete, this block passes on .true if it receives the correct input values in the correct order, and it passes on .false otherwise.

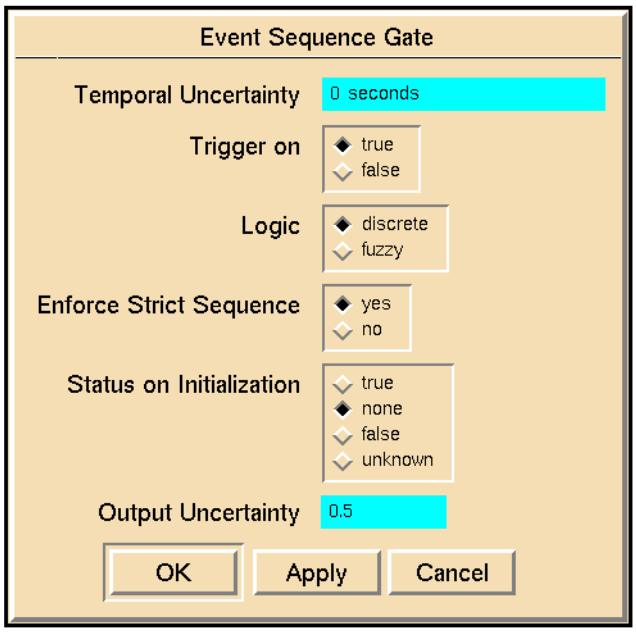
Using Fuzzy Logic

When Logic is fuzzy, this block passes on the minimum of the two input belief values if it receives the correct input values in the correct order, and it passes 0.0 otherwise.

Even though Logic is fuzzy, the block expects the attribute Trigger On to be either .true or .false, and it checks the status values of the input paths instead of the belief values. For example, when an input belief value changes from 0.90 to 0.91, the block considers the input path to have the same value as before: .true.

Configuring

This is the configuration panel for the Event Sequence Gate.

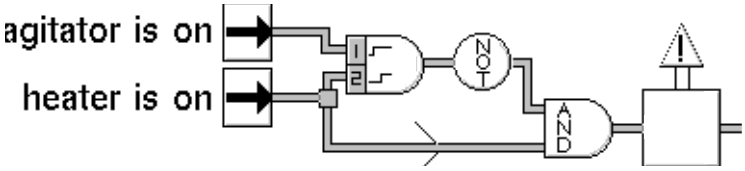


Attribute	Description
Temporal Uncertainty	The delay between the time the top input path receives the Trigger On value and the time the bottom input path receives it.
Trigger On	The truth value that the block must detect on the top input path before detecting it on the bottom input path, in order to pass a value of . true.
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.
Enforce Strict Sequence	Whether the block passes .true when it receives the Trigger On value on the bottom path exactly Temporal Uncertainty seconds after it receives the value on the bottom path (no), or whether it passes .false (yes).

Attribute	Description
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.

Example

The Event Sequence Gate in this figure raises an alarm when a tank's heater is turned on before the agitator has been on for at least 10 seconds. In other words, to avoid an alarm, you must turn on the agitator for at least 10 seconds before turning on the heater.



This table lists the values of some of the Event Sequence Gate's attributes:

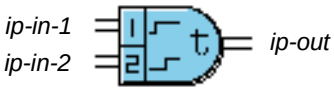
Attribute	Value
Trigger On	.true
Enforce Strict Sequence	yes
Temporal Uncertainty	10 seconds

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Synchronize block	page 136	*Reference Manual*
The True if Expired block	page 297	*Reference Manual*
The Control Synchronize block	page 461	*Reference Manual*

Event Window Gate



The Event Window Gate determines whether one of the input paths turns .true within a specified time window of the other input path turning .true. If so, the block passes a value of .true. If the second input path does not turn .true within the specified time window of the first input path turning .true, the block passes a value of .false.

The output value remains the same until the next triggering event. A triggering event occurs when the designated input path turns .true.

You can specify that either the top or the bottom input path initiates the triggering event, or you can specify that either input path triggers the event.

Specifying Which Input Triggers the Event

You specify which input triggers the event, using the Triggering Port attribute. This attribute can have one of these three values:

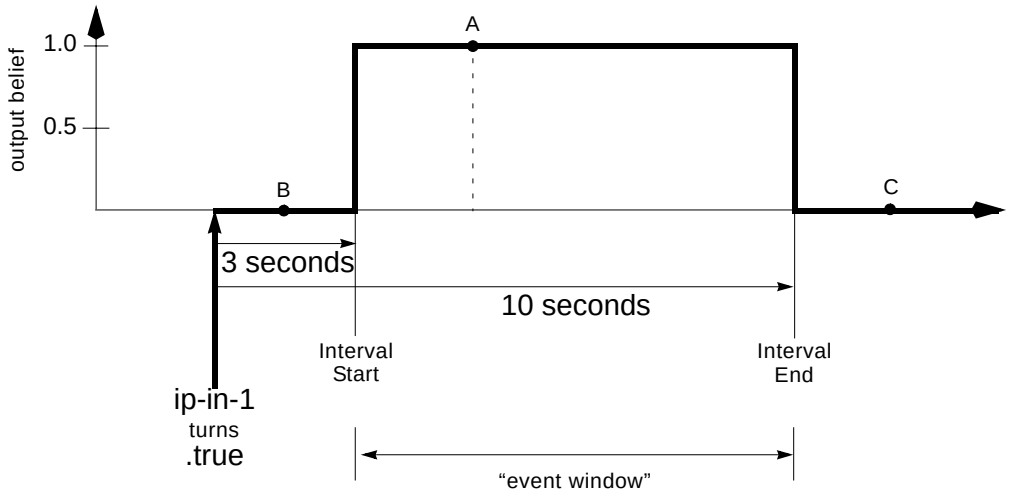
If Triggering Port is...	Then the event is triggered when...
input-1	The top input path (ip-in-1) turns .true
input-2	The bottom input path (ip-in-2) turns .true
either	Either the top or the bottom input path turns .true

Once the block is active, it disregards any new triggering input.

Specifying the Time Interval

You specify the time interval using the attributes Interval Start and Interval End. Interval Start and Interval End both specify a time period after the specified input path turns .true. Interval End must always be greater than Interval Start.

The following figure illustrates a time line showing discrete output values for an Event Window Gate given an Interval Start of 3 seconds, an Interval End of 10 seconds, and a Triggering Input of input-1. The portion of the time line where the block returns values (i.e., the membership function) appears in bold.



The following table shows the discrete status values for the labelled time periods in the figure:

If the bottom input path turns .true at time period...	Which is...	The block passes a Status-value of...	At the time of...
A	between Interval Start and Interval End	.true	Event A
B	between the time ip-in-1 turns .true and Interval Start	.false	Event B
C	greater than Interval End	.false	Interval End

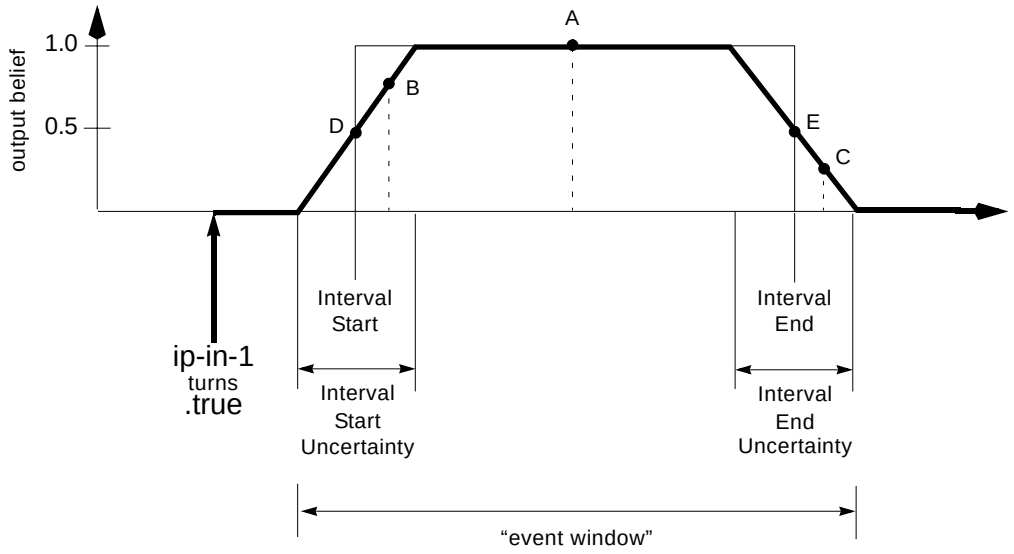
To specify that two events should happen within a particular number of seconds from each other, independent of order, set Interval Start to 0, Interval End to some number of seconds, and Triggering Input to either.

Using Fuzzy Logic

To use fuzzy logic:

1. Specify Logic as fuzzy.
2. Specify values for Interval Start and Interval End.
3. Specify values for Interval Start Uncertainty and Interval End Uncertainty.

The following diagram illustrates how the block uses these uncertainties and determines fuzzy belief values. The portion of the time line where the block returns values (i.e., the membership function) appears in bold. The discrete membership function appears using a thin line for comparison.



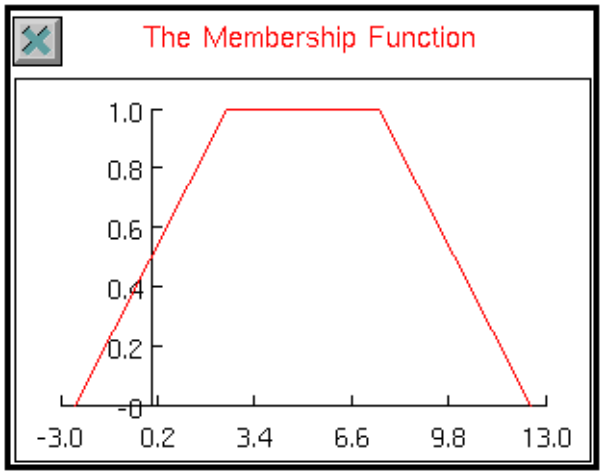
The following table shows fuzzy belief values for the labelled time periods.

If the bottom input path turns .true at time period...	**Which is...**	**The block passes a** Belief-value **of...**
A	between Interval Start + (Interval Start Uncertainty /2) and Interval End - (Interval End Uncertainty /2)	1.0
B	between Interval Start - (Interval Start Uncertainty /2) and Interval Start + (Interval Start Uncertainty /2)	scaled belief value between 0.0 and 1.0
C	between Interval End - (Interval End Uncertainty /2) and Interval End + Interval End Uncertainty /2)	scaled belief value between 0.0 and 1.0
D	Interval Start	0.5
E	Interval End	0.5

Viewing the Membership Function

When using fuzzy logic, you can view a graph of the membership function using the show membership function menu choice. This menu choice only appears if you specify a value for Interval Start Uncertainty and Interval End Uncertainty.

The following figure shows a graph of the membership function created for an Event Window gate whose Interval Start is 0, whose Interval End is 10, and whose Interval Start Uncertainty and Interval End Uncertainty are both 5 seconds.



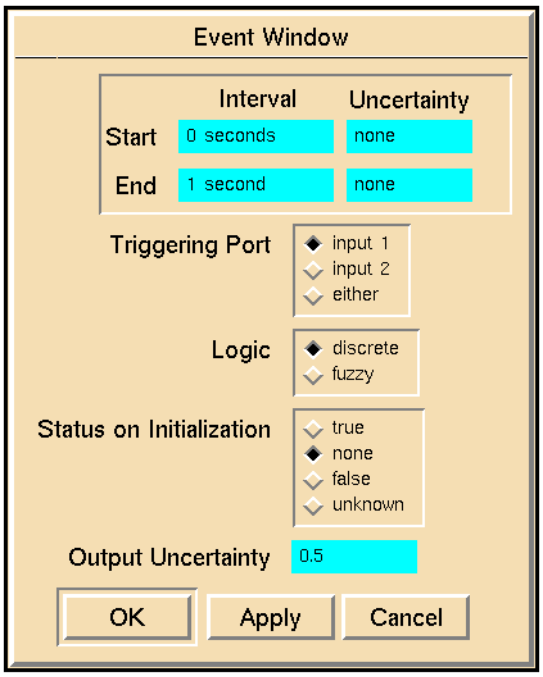
Specifying the Output Uncertainty

The Output Uncertainty determines whether the block passes a status value of .true, .false, or unknown when using fuzzy logic. Given an Output Uncertainty of 0.0, a fuzzy belief value of less than 0.5 passes .false, a fuzzy belief value of greater than 0.5 passes .true, and a fuzzy belief value of exactly 0.5 passes unknown.

Given an Output Uncertainty of greater than 0.0, the block passes unknown for a wider range of fuzzy belief values. For example, given an Output Uncertainty of 0.2, the block passes unknown for fuzzy belief values between 0.4 and 0.6, not inclusive.

Configuring

This is the configuration panel for the Event Window Gate.

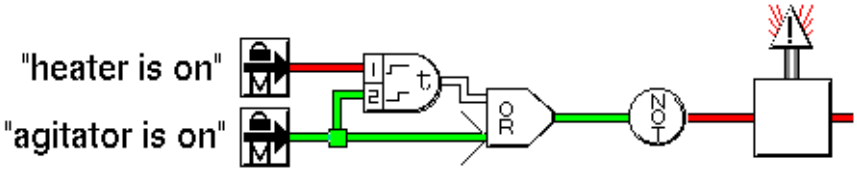


Attribute	Description
Interval Start	The start time period after the specified input path turns .true.
Interval Start Uncertainty	An uncertainty band around the Interval Start, which the block uses to determine whether an input value is within the time window.
Interval End	The end time period after the specified input path turns .true. Interval End must always be greater than Interval Start.
Interval End Uncertainty	An uncertainty band around the Interval End, which that the block uses to determine whether an input value is within the time window.
Triggering Port	The input port that must receive the triggering event first, which is a value of .true. A value of either means that the block can receive the triggering event on either input port.

Attribute	Description
Logic	See “Specifying the Type of Logic to Use” on page 101 in the *GDA User’s Guide*.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.

Example

The following diagram raises an alarm when the heater is on for more than 10 seconds without the agitator also being on. The Or Gate prevents the alarm if the agitator is turned on before the heater is turned on.



The following table lists the values for some of the Event Window Gate’s attributes.

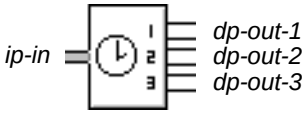
Attribute	Value
Triggering Input	input-1
Interval Start	0 seconds
Interval End	10 seconds
Output Uncertainty	0.0

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*

Inference Time Stamp



The Inference Time Stamp gate passes the last time that its input inference path had a value of .true, .false, and unknown, as shown in the following table. It also stores the times in the attributes Time of Last True, Time of Last Unknown, and Time of Last False. If the block has never received a particular status value, it passes 0 on the path for that status value.

The block passes the time its input value was last…	On the…
.true	Top port (dp-out-1)
unknown	Middle port (dp-out-2)
.false	Bottom port (dp-out-3)

Configuring

The Inference Time Stamp gate has no configurable attributes.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Time Stamp block	page 152	*Reference Manual*

14

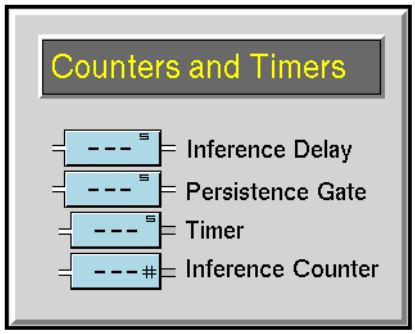
Counters and Timers

Describes the blocks that let you delay, time, and count inference values.

Introduction

The blocks on the Counters and Timers palette let you delay, time, and count inference values. The feature they all share in common is a digital readout on their icons that reflects the block's status.

You can find the Counters and Timers palette in the Inference submenu of the Palettes menu:



Pausing Values

These blocks delay their input inference values. They differ in how they handle receiving a new value when they are already delaying a value:

- The Inference Delay block on page 363 simulates a system delay, such as waiting for a valve to open fully or for a part to move from one place in the factory to another. If it is delaying a value and receives a new value, it delays them both.
- The Persistence Gate block on page 366 guarantees that its input has remained unchanged for a specified time interval before passing it. If it is delaying a value and receives a different value, it sends out the new value immediately and throws out the value it was delaying.

Timing Values

The Timer block on page 369 times how long its input inference value has a specific value. It passes that time along.

Counting Values

The Inference Counter block on page 372 counts how many times its input inference value has a specific value. It passes that count along.

See Also

The Belief Displays blocks let you show the belief value of an inference path. They are on the Path Displays palette in the Other menu.

Inference Delay

The Inference Delay block delays passing its inference input value for a specified amount of time, and displays a countdown of the time on its icon. It does not modify its input value. To specify the length of the delay, set the Delay attribute. To specify how the icon displays the countdown, set the attributes Count By and Display Units.

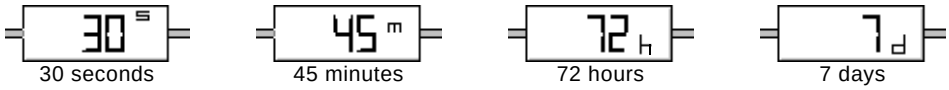
This block is useful for simulating system delays, such as:

- Fluid flowing through a pipeline, like carrying water weighed down with impurities.
- Material handling, like moving a part from one area of a factory to another.
- Mechanical delays, like waiting for a valve to move to its fully open position.

If the block is already delaying a value when it receives a new value, it continues to display the countdown for the old value. When it passes the old value, it displays the countdown for the new value, already in progress.

Specifying How to Display the Countdown

To specify how often the icon updates the countdown, set the Count By attribute. To specify the unit of time it displays, set the Display Units attribute to seconds, minutes, hours, or days. This figure shows how the Inference Delay displays 30 seconds, 45 minutes, 72 hours, and 7 days:



The icon can display numbers up to 999. If the block cannot display the time interval in the units you specified, it tries the larger units in order until it can. For example, if you set Delay to 2700 seconds and Display Units to seconds, the block displays “45 m.”

If the block needs to round the time it displays, it rounds to the nearest whole number, rounding up when it is halfway between whole numbers. For example, it displays 1 $\frac{1}{4}$ hours as “1 h,” 1 $\frac{3}{4}$ hours as “2 h,” and 1 $\frac{1}{2}$ hours as “2 h.”

Resetting

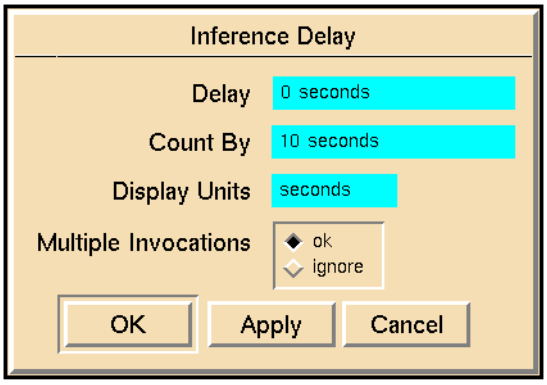
When you reset this block, it aborts any outstanding delays and passes on its input values immediately.

Receiving Simultaneous Values

Note When Multiple Invocations is `ok` and the block receives simultaneous signals, the countdown display is unpredictable.

Configuring

This is the configuration panel for the Inference Delay block.



Attribute	Description
Delay	The amount of time that the counter delays passing its input.
Count By	The frequency with which the counter updates the countdown.
Display Units	The unit of time that the counter displays.
Multiple Invocations	See "Specifying How to Handle Multiple Values" on page 92 in the *GDA User's Guide*.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

For more information on...	See...	In this book...
The Data Delay block	page 130	*Reference Manual*
The Control Delay blocks	page 441	*Reference Manual*

Persistence Gate

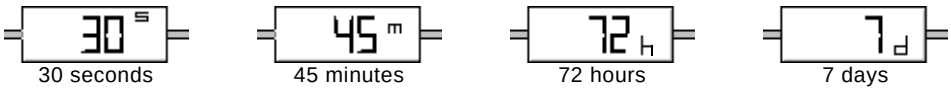
The Persistence Gate guarantees that its input value has remained unchanged for a specified time interval before passing that value. It displays a countdown of the time on its icon. To specify the time interval, set the attribute Delay. To specify how the icon displays the countdown, set the attributes Count By and Display Units.

To delay a specific status value, set the attribute Trigger On to .true, .false, or unknown. To delay all status values, set Trigger On to always.

When the block receives the value Trigger On, it begins a countdown and passes the value when the countdown reaches 0. If the block receives a different status value during the countdown, it aborts the countdown and passes the new value immediately. If the block receives the status Trigger On again during a countdown, it continues the countdown without interruption.

Specifying How to Display the Countdown

To specify how often the icon updates the countdown, set the Count By attribute. To specify the unit of time it displays, set the Display Units attribute to seconds, minutes, hours, or days. This figure shows how the Persistence Gate displays 30 seconds, 45 minutes, 72 hours, and 7 days:



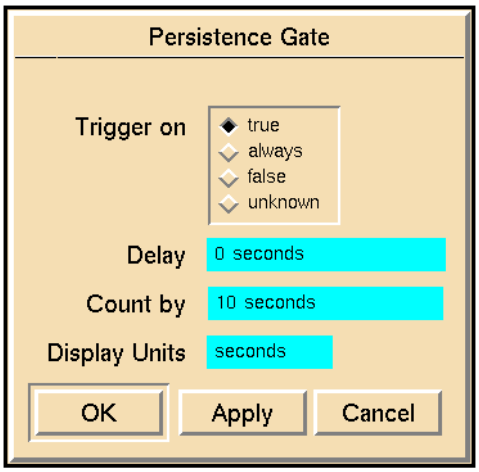
The icon can display numbers up to 999. If the block cannot display the time interval in the units you specified, it tries the larger units in order until it can. For example, if you set Delay to 2700 seconds and Display Units to seconds, the block displays “45 m.”

If you specify a value for the Delay or Count By attributes without specifying the units, GDA assumes the units to be seconds and converts the value in the configuration panel to minutes and seconds.

If the block needs to round the time it displays, it rounds to the nearest whole number, rounding up when it is halfway between whole numbers. For example, it displays 1 $\frac{1}{4}$ hours as “1 h,” 1 $\frac{3}{4}$ hours as “2 h,” and 1 $\frac{1}{2}$ hours as “2 h.”

Configuring

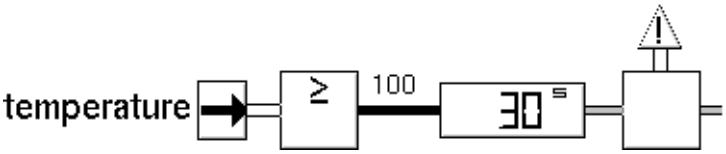
This is the configuration panel for the Persistence Gate.



Attribute	Description
Trigger On	The specific status value to delay. A value of always delays any status value that the gate receives.
Delay	The amount of time that the gate delays passing its input.
Count By	The frequency with which the gate updates the countdown.
Display Units	The unit of time that the gate displays.

Example

The diagram in this figure raises an alarm whenever the temperature is over 100° for more than 30 seconds. In the diagram, the entry point just passed a temperature of 115° and the Persistence Gate is counting down from 30 seconds.



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Delay block	page 130	*Reference Manual*
The Control Delay blocks	page 441	*Reference Manual*

Timer

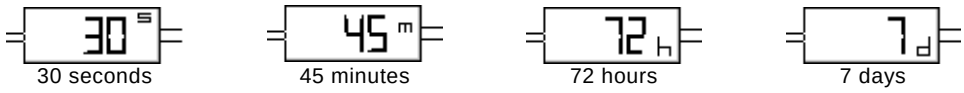
The Timer block logs the amount of time that its input path has a specified value. The block begins counting when the input value matches the attribute Trigger On, and it stops counting when the input value matches the attribute Stop When. To specify how often the block updates its readout and passes on the time interval, set the attribute Count By. To specify the unit of time it displays, set the attribute Display Units.

You can control whether the Timer's log is cumulative or not. When the input value changes to Trigger On, the attribute Accumulate Values determines what happens next. If it is no, the block resets its log to 0. If it is yes, the block continues counting from where it was before.

If the timer receives a value other than Trigger On or Stop When while it is counting, it ignores the input.

Specifying How to Display the Countdown

To specify the unit of time it displays, set the Display Units attribute to seconds, minutes, hours, or days. This figure shows how the Timer displays 30 seconds, 45 minutes, 72 hours, and 7 days:

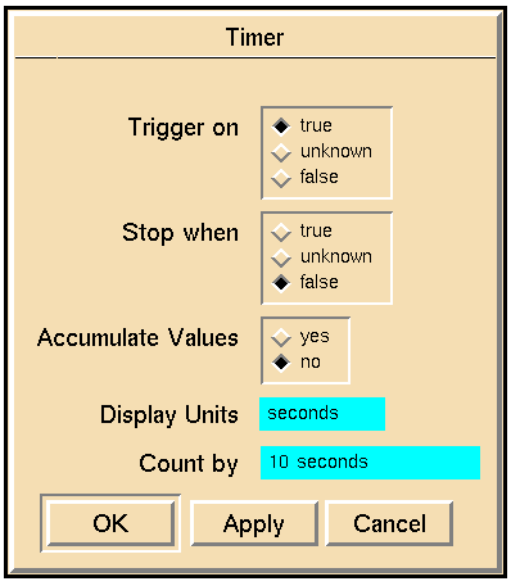


The icon can display numbers up to 999. If the block cannot display the time interval in the units your specified, it tries the larger units in order until it can. For example, if you set Delay to 2700 seconds and Display Units to seconds, the block displays "45 m."

If the block needs to round the time it displays, it rounds to the nearest whole number, rounding up when it is halfway between whole numbers. For example, it displays 1 $^{1}/_{4}$ hours as "1 h," 1 $^{3}/_{4}$ hours as "2 h," and 1 $^{1}/_{2}$ hours as "2 h."

Configuring

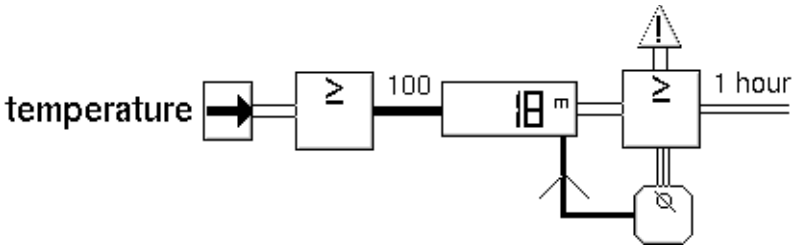
This is the configuration panel for the Timer block.



Attribute	Description
Trigger On	The value that starts the timer.
Stop When	The value that stops the timer.
Accumulate Values	Whether the timer resets to 0 when the input value changes to the Trigger On value (no), or whether the timer continues counting (yes).
Display Units	The unit of time that the counter displays.
Count By	The frequency with which the counter updates the countdown.

Example

The diagram in this figure times how long the tank's temperature stays above 100°. When the temperature has been over 100° for over an hour, it raises an alarm and resets the Timer. The alarm might tell the operator to perform some maintenance on the tank to repair the harm the high temperature caused.



Here are the settings of some of the Timer's attributes:

Attribute	Value
Display Units	minutes
Count By	1 minute
Trigger On	.true
Stop When	.false
Accumulate Values	yes

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Inference Counter

The Inference Counter counts the number of times its input inference path changes to a specified status value. The block displays the count on its icon and passes the count as its output. As long as the input path has the Trigger On value, the block does not increase its count.

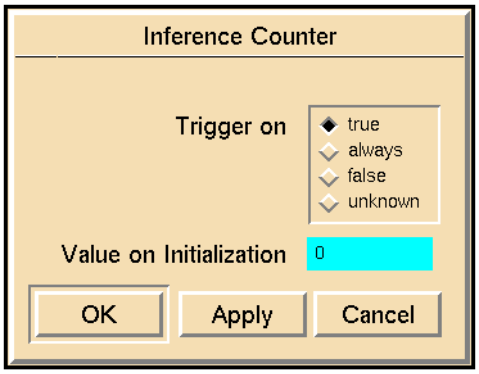
To count a particular status value, set the attribute Trigger On to .true, .false, or unknown. To count the number of times the input path changes from one value to any other value, set Trigger On to always.

If the block receives fuzzy belief values, it increments its count each time it receives a new belief value that has the status value in Trigger On. For example, if Trigger On is .true and the block receives the belief values 0.80, 0.90, and 0.95, it increments its count 3 times.

To set the count back to the Value on Initialization, reset the block.

Configuring

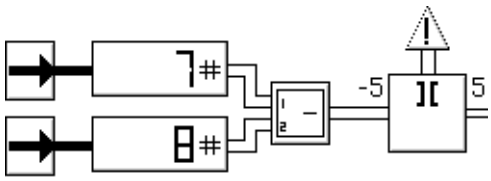
This is the configuration panel for the Inference Counter.



Attribute	Description
Trigger On	The input value that the block counts. A value of always causes the block to count the number of times the input value changes from one inference value to another.
Value on Initialization	The value of the counter upon reset. The value must be an integer. See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Example

The diagram in this figure raises an alarm if one of the entry points becomes .true five times more than the other entry point:



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying an Initial Data Value	page 83	*User's Guide*
The Input Counter block	page 150	*Reference Manual*

15

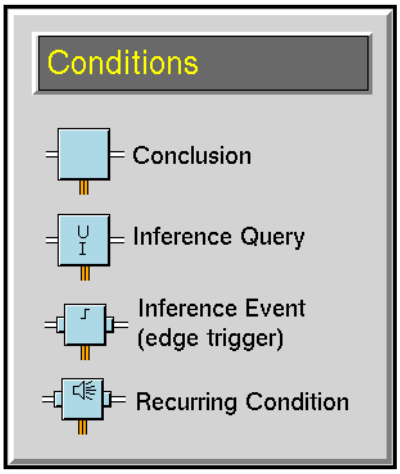
Conditions

Describes the blocks that connect to blocks on the Observations and Conditions palette, which send control signals that trigger various actions.

Introduction

The blocks on the Conditions palette provide you with special properties that most other inference blocks do not have. The properties include raising alarms, triggering action blocks, and producing explanations. To keep the inference blocks as simple as possible, these properties are restricted to the blocks on the Observations and Conditions palette.

You can find the Conditions palette in the Inference submenu of the Palettes menu:



These blocks provide you with these additional properties:

- **Raising alarms.** You can attach Alarm and Recurring Alarm capabilities, described on page 519, to them.
- **Producing explanations.** They have the Description When Unknown, Description When True, and Description When False attributes, described in "How to Describe Why a Block Passes True, False, or Unknown" on page 94 in the *GDA User's Guide*, which let you provide full explanations of the condition. The descriptions can include the values of expressions, as described in "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*. You can also attach capabilities that affect explanations, including Explanation Memory, Local Explanation Restriction, and Ignore Path Explanation Restriction.
- **Providing hysteresis.** You can ignore insignificant changes in the belief value by setting the Hysteresis When attribute, described in "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.
- **Triggering action blocks.** Every condition has an output Control Path on the bottom of its icon. The block passes a control signal down that path whenever it passes a .true status value. For more information, see "Using Control Paths" on page 63 in the *GDA User's Guide*.
- **Triggering capabilities.** You can attach any capability to a Condition, including those dealing with alarms. For more information, see "Capabilities and Restrictions" on page 515.
- **Overriding.** Every condition has an override menu choice that lets you manually enter the value it passes on, described in "Overriding Block Values" on page 51 in the *GDA User's Guide*.

- **Locking.** Every condition also has a lock menu choice, described in "Locking and Unlocking Blocks" on page 55 in the *GDA User's Guide*, that lets you lock in the value. The output value will not change until you choose the unlock menu choice.

Drawing Conclusions

The Conclusion block on page 378 is the simplest condition block. It serves no function other than providing you with the condition block's special properties.

Querying the Operator

The Inference Query block on page 381 lets you ask the operator to enter a status value. It displays a dialog with two or three radio buttons, one for each status value: .true, .false, and sometimes unknown.

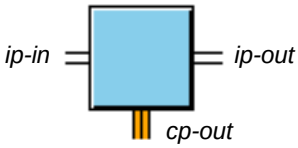
Detecting Status Changes

The Inference Event (Edge Trigger) block on page 384 detects when an inference path changes its value to a certain value. It is also called an **edge trigger**, since it passes .true when it passes an **edge**: from .false to .true or from .true to .false.

Detecting Recurring Conditions

The Recurring Condition block on page 387 detects when its input path receives a certain value repeatedly. It is useful when you want to raise an alarm when a change happens frequently, but you want to ignore infrequent, isolated changes.

Conclusion



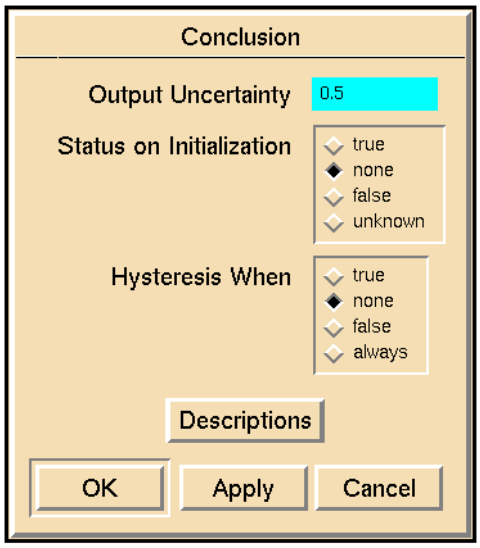
The Conclusion block lets you perform these actions after drawing a conclusion:

- Activate action blocks.
- Trigger alarms and other capabilities.
- Provide descriptions for explanations.
- Override the path's value with the override menu choice.

It is especially useful after blocks that do not connect to action blocks, and capabilities, such as the blocks in the Logic Gates, Tabular Gates, Evidence Gates, and Temporal Gates palettes. It also lets you define a new uncertainty band or a different type of hysteresis without performing any additional inference analysis.

Configuring

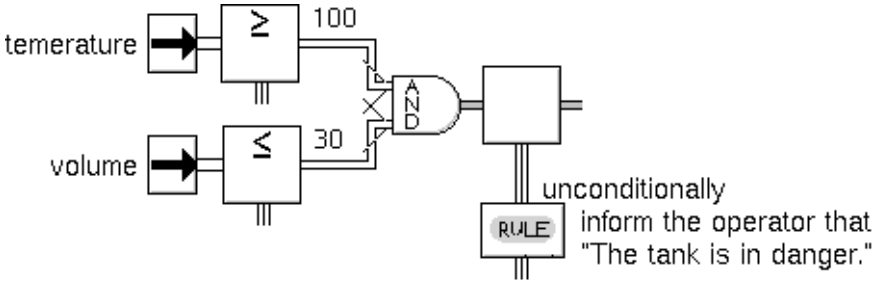
This is the configuration panel for the Conclusion block.



Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Hysteresis When	See “Specifying Hysteresis” on page 103 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

Example

This figure shows a Conclusion block connected to an AND Gate that concludes whether both the temperature of a tank is high and the volume of the tank is low. The Conclusion block has several capabilities and action blocks attached to it, which cannot be attached to the AND Gate. The Capabilities raise an alarm if the Conclusion is .true and remembers the explanation for the last .true input. The Action blocks play a warning sound and activate a subworkspace of rules to diagnose the problem.

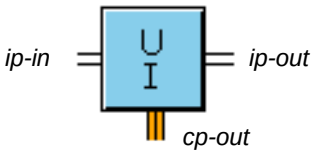


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying Fuzzy Logic Attributes	page 101	*User's Guide*

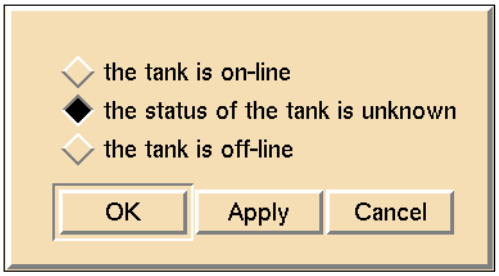
Inference Query



The Inference Query block lets the operator enter an inference value. When its input status value is .true, it displays an override dialog (similar to the override dialog) that has two or three radio buttons, one for each status value: .true, .false, and sometimes unknown. To choose the descriptions of the dialog's radio buttons, set the attribute Description When True, Description When False, and Description When Unknown. The block passes the value you choose on its output path. If the block's input value is .false or unknown, the block does nothing.

Note To add a prompt to the Inference Query block's dialog, use the Dialog Restriction block on page 555.

If the Output Uncertainty attribute is none, the block's dialog does not contain a radio button for unknown and contains only two buttons. Otherwise it contains three buttons, as in the figure below. In the example, Description When True is "The tank is on-line", Description When False is "The tank is off-line", and Description When Unknown is "The status of the tank is unknown".



If you press OK, the block passes the value you chose, and the dialog disappears. If you press Apply, the block passes the value you chose, and the dialog remains on the screen. If you press Cancel, the block passes nothing, and the dialog disappears.

Specifying the G2 Windows in Which the Dialog Appears

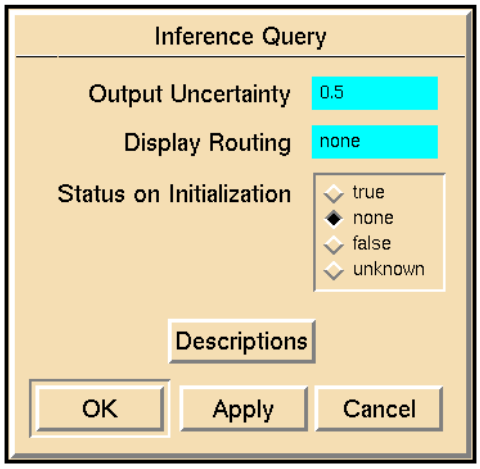
To specify which G2 windows the dialog should appear on, use the Display Routing attribute. It can have these values:

If Display Routing is...	The dialog appears in these G2 windows...
A Display Routing object	All the windows specified in the object
A Remote Window object	The G2 window specified in the object
none	All G2 windows

For more information, see the Display Routing block on page 594 and the Remote Window block on page 592.

Configuring

This is the configuration panel for the Inference Query block.

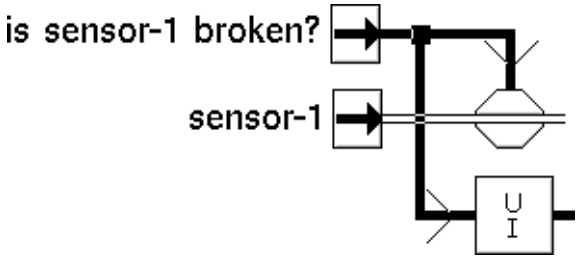


Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Display Routing	The G2 window on which the dialog appears.

Attribute	Description
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

Example

The Inference Query block in this figure lets the operator enter a value for sensor-1 if the sensor is broken. When sensor-1 is broken, the Inference Inhibit block stops the sensor-1’s value from being passed and the Inference Query block is activated.

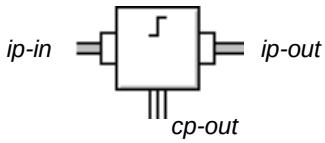


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Overriding Block Values	page 51	*User’s Guide*
Specifying and Generating Explanations	page 93	*User’s Guide*
Evaluating Expressions in Attributes	page 118	*User’s Guide*
Specifying Fuzzy Logic Attributes	page 101	*User’s Guide*
Remote Window	page 592	*Reference Manual*
Display Routing	page 594	*Reference Manual*

Inference Event (Edge Trigger)



The Inference Event block detects when an inference path changes its value to a certain value and holds that value for a certain amount of time. To detect when the input path changes to a specific status value, set Trigger On to .true, .false, or unknown. To detect when the input path changes to any status value, set Trigger On to always.

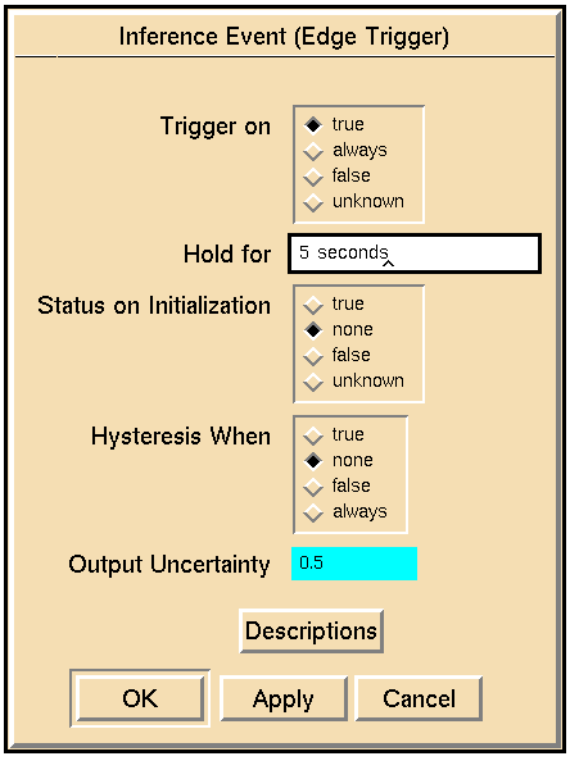
The block is also called an **edge trigger**. If you call a change from .true to .false a falling edge and a change from .false to .true a rising edge, the block passes .true when it detects a falling or rising edge.

When the block receives a value specified by Trigger On, it passes the status value .true, holds that value for Hold For seconds, and then passes the status value . false. If the block receives the value Trigger On while it is still holding another value, it starts waiting again and passes .false after Hold For seconds.

If Hold For is none, the block passes .true and .false in quick succession. This momentary transition to .true is called a "pulse."

Configuring

This is the configuration panel for the Inference Event block.



Attribute	Description
Trigger On	The truth value that the block detects as the "edge." A value of always causes the block to detect any status value.
Hold For	The amount of time that the block passes a value of .true after it detects the edge.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Hysteresis When	See "Specifying Hysteresis" on page 103 in the *GDA User's Guide*.

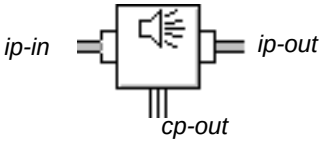
Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Overriding Block Values	page 51	*User’s Guide*
Specifying and Generating Explanations	page 93	*User’s Guide*
Evaluating Expressions in Attributes	page 118	*User’s Guide*
Specifying Fuzzy Logic Attributes	page 101	*User’s Guide*

Recurring Condition



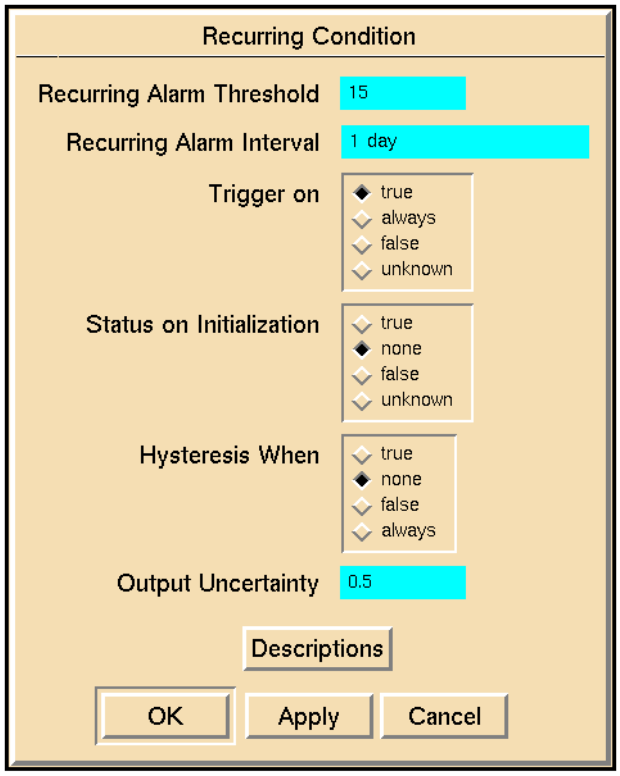
The Recurring Condition block detects when its input path changes to a certain value repeatedly. The block passes .true when it changes to a specified status value a certain number of times during a specified amount of time. You specify the number of times the block receives the value in the Recurring Alarm Threshold attribute, and you specify the amount of time that the block tests the inputs for the Trigger On value in the Recurring Alarm Interval attribute. If the block receives a status value that is the same as the last status value it received, it does not count that as a change.

This block is especially good at raising an alarm when a change happens frequently while ignoring infrequent isolated changes. For example, this block could determine whether a sensor is broken by checking how often it returns out-of-range values.

To detect a particular status value, set Trigger On to .true, .false, or unknown. To detect any change in the status value, set Trigger On to always. To specify the number of times the input path must change to that value, set the attribute Recurring Alarm Threshold to that number. To specify the time interval, set the attribute Recurring Alarm Interval.

Configuring

This is the configuration panel for the Recurring Condition block.

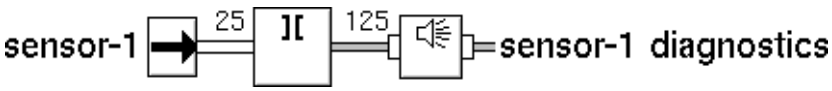


Attribute	Description
Recurring Alarm Threshold	The number of times that the block must receive the Trigger On value before it passes a value of .true.
Recurring Alarm Interval	The amount of time that the block tests the inputs for the Trigger On value.
Trigger On	The truth value that the block detects. A value of always causes the block to detect any status value.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.
Hysteresis When	See “Specifying Hysteresis” on page 103 in the *GDA User’s Guide*.

Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

Example

In this figure, a Recurring Condition block keeps track of how often a sensor goes out of range (25 to 125). When the sensor goes out of range more than 5 times in an hour, the Recurring Condition block diagnoses the sensor to see if it is broken.



Here are the settings of some of the attributes for the Recurring Condition block in the example:

Attribute	Value
Recurring Alarm Threshold	5
Recurring Alarm Interval	1 hour

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Overriding Block Values	page 51	*User’s Guide*
Specifying and Generating Explanations	page 93	*User’s Guide*
Evaluating Expressions in Attributes	page 118	*User’s Guide*
Specifying Fuzzy Logic Attributes	page 101	*User’s Guide*

Part III

Action Blocks

16

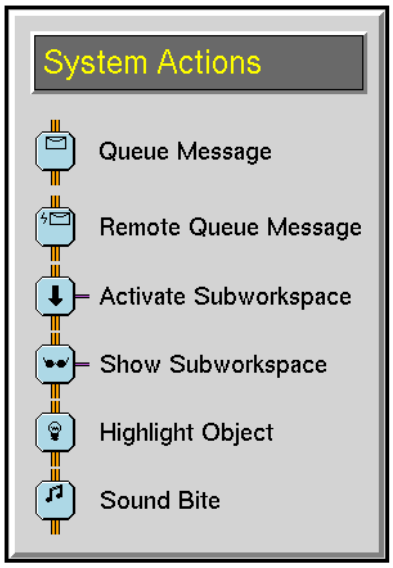
System Actions

Describes the blocks that perform actions on GDA blocks and other objects.

Introduction

Action blocks, which are also called "control blocks," let you perform a variety of actions on objects in the G2 environment, including GDA blocks. The System Actions, described in this chapter, let you operate on queues, subworkspaces, G2 icons, and sounds. Action blocks in other chapters help you form sequences of actions, let you operate on GDA blocks, and let you ask the operator questions.

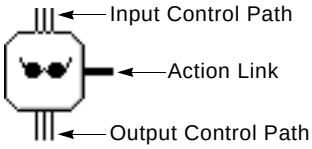
You can find the System Actions palette under the Action submenu of the Palettes menu:



System Action Block Paths

You connect Action blocks together with Control Paths, which carry control signals. A block receives a control signal, performs its action, passes the control signal to the next block, and so on. A control signal does not have a value, but it makes sure that an action block is evaluated only after the previous one is done.

Unlike other blocks, an Action block's paths are on the top and bottom: the input path is on top and the output path is on the bottom, as in the following figure. This difference lets you connect an Observation block to both an Action block and an Inference block: the Action block connects on the bottom and the Inference block connects on the right. When the Observation is .true, it sends a control signal down its Control Path, in addition to sending a belief value along its Inference Path.



Some Action blocks also have an Action Link, shown in the figure above, which you connect to another GDA block. When the Action block is evaluated, it performs its action on any block connected to its Action Link. For example, the Show Subworkspace block displays the subworkspace of any block connected to its Action Link.

Sending Messages

These blocks let you send messages to user queues:

- The Queue Message block on page 396 posts a new message to a queue.
- The Remote Queue Message block on page 400 posts a message to a queue on a remote G2 process.

Working with Subworkspaces

These blocks let you work with subworkspaces:

- The Activate Subworkspace block on page 404 activates subworkspaces, letting G2 evaluate all the rules on them.
- The Show Subworkspace block on page 407 displays a subworkspace. It is useful when you need to display documentation or instructions.

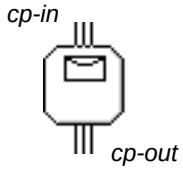
Highlighting Objects

The Highlight Object block on page 411 highlights a G2 object by changing the colors for some of its regions.

Playing Sounds

The Sound Bite block on page 415 plays sounds on Sun and Hewlett-Packard workstations. It is useful when you want to alert the operator with an alarm sound or play a voice message.

Queue Message



The Queue Message block posts a new message to a queue every time it evaluates. Specify the text of the message in the attribute Entry Text and the name of the queue in the attribute Queue Name.

Specifying the Queue

Enter the name of the queue that the message will appear on in the attribute Queue Name. By default, it is message-queue. If Queue Name is none, GDA does not display your message and passes its control signal. If Queue Name contains the name of a queue that does not exist, GDA writes an error message to the logbook and passes its control signal.

If you want GDA to bring the queue to the front after it posts your message, set the attribute Show Queue to yes. Otherwise, set Show Queue to no.

Specifying the Message

Specify the text of the message in the attribute Entry Text. The rest of this section describes how to specify other features of the message, such as its severity.

Setting the Severity

To let the operator know how severe the alarm is, set the attribute Entry Priority. Entry Priority can be any number from 1 to 15, where 1 is most severe and 15 is least severe.

Each severity level has a color associated with it. GDA uses that color as the background color for an unacknowledged message entry in a queue. The default colors for the severity levels are as follows: 1 is red, 2 is orange, 3 is gold, and the rest are gray. To change the colors of the alarms, select Preferences > Colors > Alarms, and edit the colors on the Alarm Colors dialog.

Deleting Unacknowledged Messages

Generally, you have to acknowledge a message before you can delete it from the queue with the Delete Entries button. If you want to delete messages without acknowledging them first, set the attribute Require acknowledgement to no.

Note that when Require acknowledgement is yes, GDA can still delete unacknowledged messages in some cases. If you press the Flush Queue button in

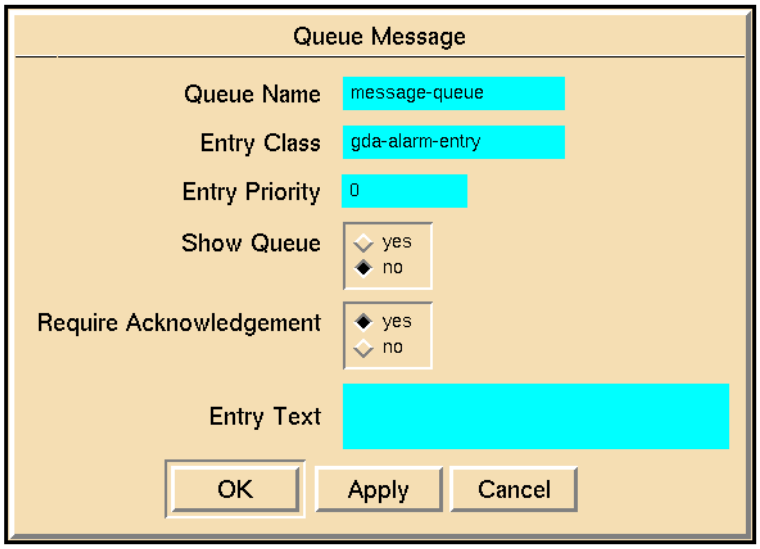
a message queue, the queue deletes all its messages, even unacknowledged ones. And if the number of messages in a queue equals its Maximum Number of Entries attribute and it receives a new message, it deletes the oldest message, even if it is unacknowledged.

Setting the Type of Queue Entry

GDA uses a G2 class to create messages in a queue. If you have created your own class for alarm messages and want an alarm to use it, set the attribute Entry Class to the name of your class. By default, it is gda-alarm-entry.

Configuring

This is the configuration panel for the Queue Message block.

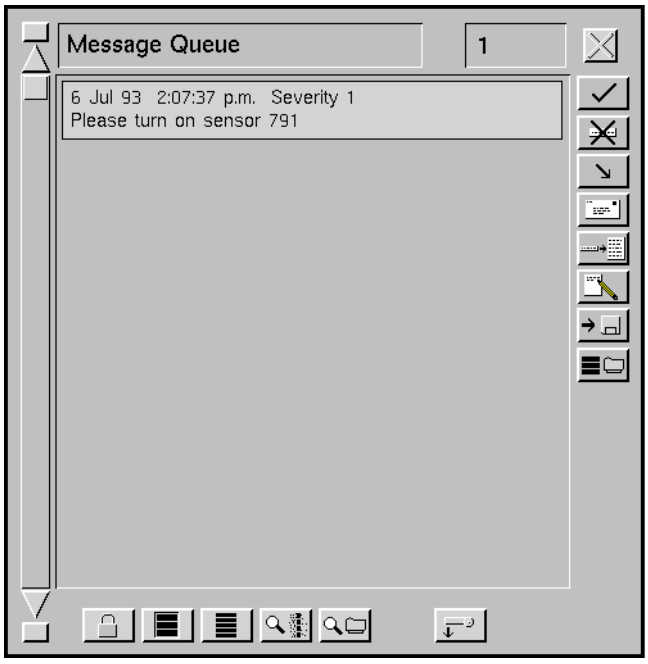


Attribute	Description
Queue Name	The name of the queue on which to display the message.
Entry Class	The class of message to appear in the queue. This attribute is obsolete.
Entry Priority	A number from 1 to 15 that determines the severity of the message and its color.
Show Queue	Whether GDA displays the queue when it receives a message or not.

Attribute	Description
Require acknowledgement	Whether a message must be acknowledged before it can be deleted (yes) or not (no).
Entry Text	The text of the message to display in the queue.

Example

The following figure shows a message that a Queue Message block created:



The block has these attributes:

Attribute	Value
Queue Name	message-queue
Entry Class	gda-alarm-entry
Show Queue	yes
Entry Text	"Please turn on sensor 791"

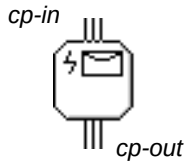
Attribute	Value
Entry Priority	1
Require acknowledgement	yes

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Alarm, Recurring Alarm Capabilities	page 519	*Reference Manual*

Remote Queue Message



The Remote Queue Message block posts a new message to a queue on a remote G2 process running GDA every time it is evaluated. Specify the text of the message in the attribute Entry Text and the name of the queue in the attribute Queue Name.

Specifying the Queue

Enter the name of the queue that the message will appear on in the attribute Queue Name. By default, it is message-queue. If Queue Name is none, GDA does not display your message and passes its control signal. If Queue Name contains the name of a queue that does not exist, GDA writes an error message to the logbook and passes its control signal.

Specifying the Message

Specify the text of the message in the attribute Entry Text. The rest of this section describes how to specify other features of the message, such as its severity.

Setting the Severity

To let the operator know how severe the alarm is, set the attribute Entry Priority. Entry Priority can be any number from 1 to 15, where 1 is most severe and 15 is least severe.

Each severity level has a color associated with it. GDA uses that color as the background color for an unacknowledged message entry in a queue. The default colors for the severity levels are as follows: 1 is red, 2 is orange, 3 is gold, and the rest are gray. You can change the colors with the Alarm Colors palette. To use the Alarm Colors palette, bring up the Configuration Tools menu bar, go to the Colors menu, and choose Alarm Colors.

Deleting Unacknowledged Messages

Generally, you have to acknowledge a message before you can delete it from the queue with the Delete Entries button. If you want delete messages without acknowledging them first, set the attribute Require acknowledgement to no.

Note that when Require Acknowledgement is yes, GDA can still delete unacknowledged messages in some cases. If you press the Flush Queue button in

a message queue, the queue deletes all its messages, even unacknowledged ones. And if the number of messages in a queue equals its Maximum Number of Entries attribute and it receives a new message, it deletes the oldest message, even if it is unacknowledged.

Setting the Type of Queue Entry

GDA uses a G2 class to create messages in a queue. If you have created your own class for alarm messages and want an alarm to use it, set the attribute Entry Class to the name of your class. By default, it is gda-alarm-entry.

Specifying the Remote G2 Process

To specify the G2 processes the message will appear on, use the attribute G2 Process. You can set it to the name of a Remote G2 Process object, all-hosts, all-remote-hosts, or none:

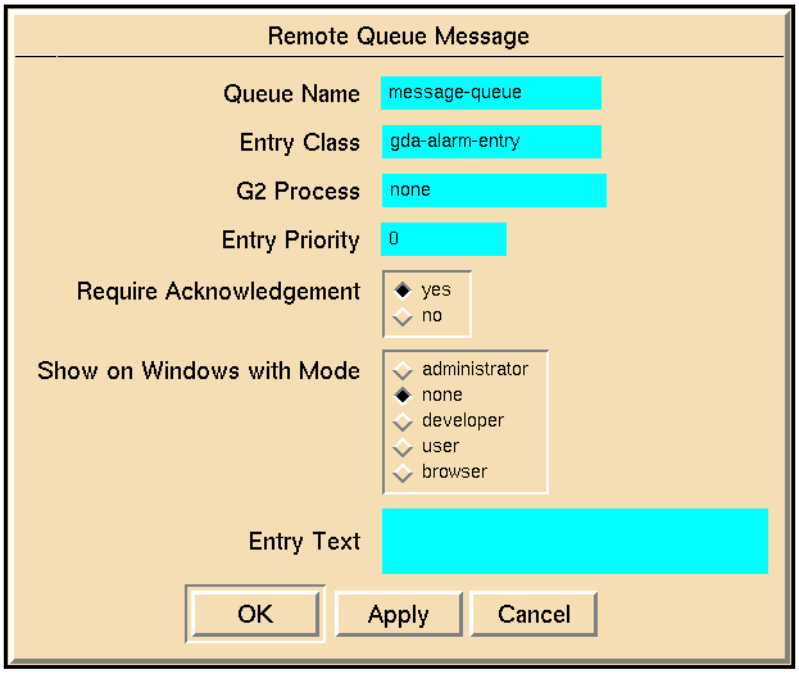
If G2 Process is...	The block prints the message to…
A Remote G2 Process name	The Remote G2 Process with that name.
all-hosts	All Remote G2 Processes and the local G2 process.
all-remote-hosts	All Remote G2 Processes.
none	No G2 process. The message is not displayed anywhere.

Note Remote G2 Process is an object on the Network palette in the Other Tools menu. It lets you give a name to a G2 process that is running on another machine on a network.

To specify which user modes the message will appear in, use the attribute Show on Windows with Mode. The user mode for a G2 window is stored in the G2-user-mode attribute, which you set by using the Main Menu > Change Mode menu choice. Some common modes are administrator and developer. If Show on Windows with Mode is none, the message will appear in all G2 windows. Otherwise, the message appears in a G2 window only if its user mode matches the value of this attribute.

Configuring

This is the configuration panel for the Remote Queue Message block.



Attribute	Description
Queue Name	The name of the queue on which to display the message.
Entry Class	The class of message to appear in the queue. This attribute is obsolete.
G2 Process	The name of the G2 process(s) on which the message queue displays.
Entry Priority	A number from 1 to 15 that determines the severity of the message and its color.
Require acknowledgement	Whether a message must be acknowledged before it can be deleted (yes) or not (no).

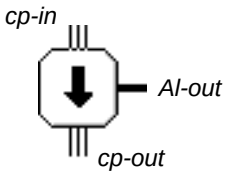
Attribute	Description
Show on Windows with Mode	The user mode in which the message queue can appear. The message queue appears only in those G2 processes that use this user mode.
Entry Text	The text of the message to display in the queue.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Alarm, Recurring Alarm Capabilities	page 519	*Reference Manual*

Activate Subworkspace



The Activate Subworkspace block activates its own subworkspace and the subworkspaces of other objects. It is especially useful for activating G2 initially rules, which can, in turn, trigger other G2 rules and procedures. In most cases, you put your initially rules on the block's subworkspace and the block will fire them when it is evaluated.

Once this block activates a subworkspace, that subworkspace remains activated until you reset this block. If the subworkspace contains rules you want to fire each time the block is evaluated, you must reset the block after it is evaluated.

To let you know that the block's subworkspaces are active, the arrow on the block's icon changes from black to red after the block is evaluated. The arrow remains red even after the block passes on the control signal. When you reset the block, the arrow changes back to black.

Specifying Which Workspaces to Activate

This block activates subworkspaces in two steps:

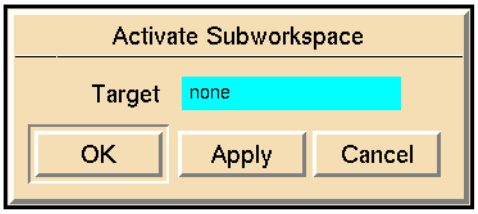
1. First, it activates the subworkspace of the block named in the Target attribute. If Target is none, it activates its own subworkspace. If it has no subworkspace, it does not activate a subworkspace in this step.
2. Next, it activates the subworkspaces of the blocks connected to its action link.

Note The object definition for the object whose subworkspace you are trying to activate must declare a configuration, using activatable-subworkspace.

The block does not signal an error if there are no subworkspaces for blocks connected with an action link. However, it does signal an error if there is no subworkspace for a block named in the Target attribute.

Configuring

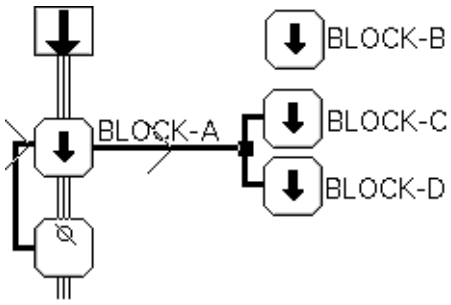
This is the configuration panel for the Activate Subworkspace block.



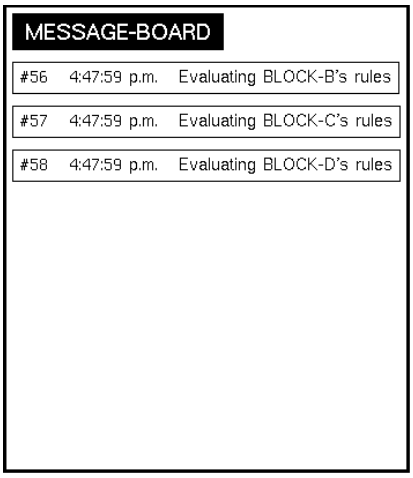
Attribute	Description
Target	The name of the block whose subworkspace is activated.

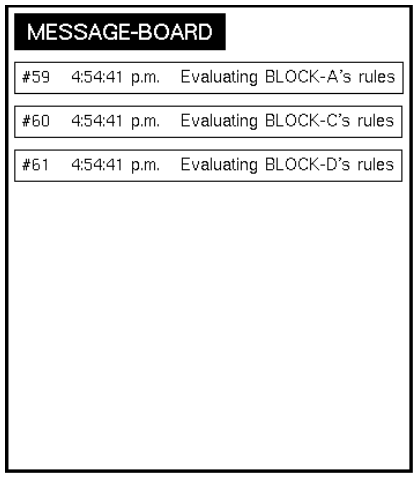
Example

This figure shows a diagram with an Activate Subworkspace block, named Block-A. It is connected to a Reset Block block, which resets it each time it is evaluated, so the initially rules for the Activate Subworkspace's subworkspaces will be evaluated each time the block is evaluated.



The Activate Subworkspace block has its own subworkspace and is connected to two other blocks with subworkspaces: Block-C and Block-D. Block-B has a subworkspace and can be the Target for the block. Each subworkspace has an initially rule which writes a message to the Message Board. The following figure shows the results of evaluating the Activate Subworkspace block. In the first message board, the Target of Block-A is Block-B, so Block-B's subworkspace was activated instead of Block-A's. In the second message board, the Target of Block-A is none, so Block-A's subworkspace was activated instead of Block-B's.



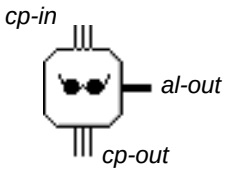


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Arithmetic Function block	page 122	*Reference Manual*
The Generic Action block	page 463	*Reference Manual*

Show Subworkspace



The Show Subworkspace block displays its own subworkspace and the subworkspaces of other objects. You can choose the location the subworkspaces are displayed at and the windows on which they are displayed. This block is especially useful when you need to display online documentation or instructions.

To let you know when the block is displaying subworkspaces, the eyeglasses on the icon change from black to white after the block is evaluated. The eyeglasses remain white even after the block passes the control signal and after you hide all the subworkspaces it showed. They change back to black only when the block is reset.

Specifying Which Workspaces to Show

The Show Subworkspace block shows subworkspaces, using the following technique:

1. First, it shows the subworkspace of the block named in the Target attribute. If Target is none, it shows its own subworkspace. If it has no subworkspace, it does not show a subworkspace.
2. Next, it shows the subworkspace of the block connected to its action link.

The block does not signal an error if there are no subworkspaces for the block connected to its action link. It does signal an error, however, if there is no subworkspace for the block named by the Target attribute.

Specifying the Location of the Subworkspace

You can specify where on the screen the subworkspaces appear with the attribute Screen Position, which can have these values:

If Screen Position is...	The subworkspaces are shown in this part of the screen...
center	Center
top-left	Top-left corner
top-right	Top-right corner

If Screen Position is...	The subworkspaces are shown in this part of the screen...
bottom-left	Bottom-left corner
bottom-right	Bottom-right corner
none	The same position it was in when it was last shown

If you specify one of the four corners of the screen, the block puts the subworkspace as far into the corner as possible while still showing all of it.

If you are displaying more than one subworkspace with this block, set Screen Position to none and position the subworkspaces yourself. Otherwise, this block places them one on top of the other.

Specifying the G2 Windows in which the Subworkspaces Appear

To specify which G2 windows the subworkspace should appear in, use the Display Routing attribute, which can have these values:

If Display Routing is...	The subworkspaces appear in these G2 windows...
A Display Routing object	All the windows specified in the object
A Remote Window object	The G2 window specified in the object
none	All G2 windows

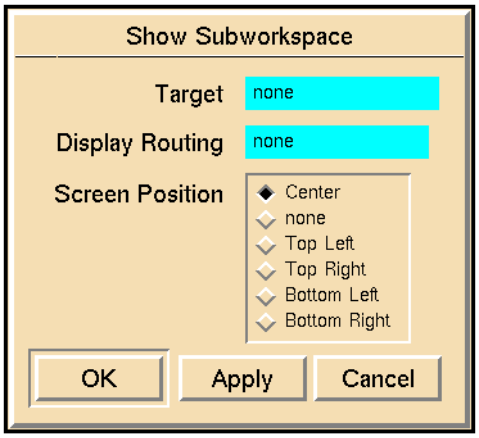
For more information, see the Display Routing block on page 594 and the Remote Window block on page 592.

Resetting

When you reset a Show Subworkspace object, it hides all the workspaces that it displayed.

Configuring

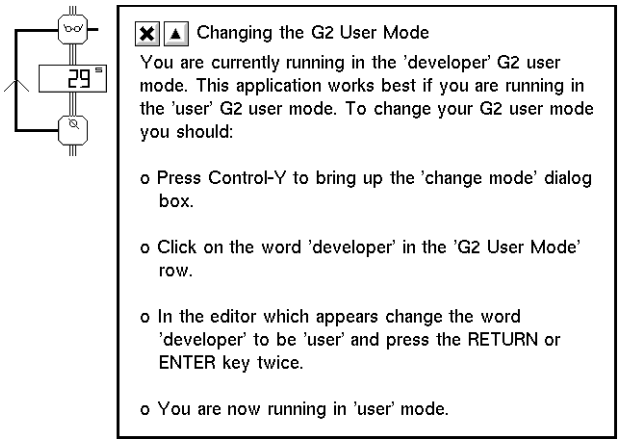
This is the configuration panel for the Show Subworkspace block.



Attribute	Description
Target	The name of the block whose subworkspace is shown.
Display Routing	The G2 windows in which the subworkspace is shown.
Screen Position	The location of the subworkspace that is shown.

Example

In this figure, a Show Subworkspace block is displaying instructions. The block is followed by a Control Delay block (with a Delay of 30 seconds) and a Reset Block, so this diagram will display the instructions for 30 seconds and then hide them again. Notice that the eyeglasses on the Show Subworkspace icon are unfilled to show that it is displaying a subworkspace.

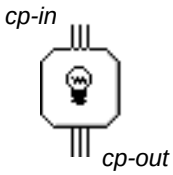


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Highlight Object



The Highlight Object block highlights these icon regions in a G2 object: icon-background, outline, status, and alarm. Specify the name of the object to highlight in the attribute Target. It gives you one way to animate a G2 diagram from within a GDA diagram.

Note You can set Target to an expression that evaluates to an object. For more information, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.

Setting the Colors

To choose the highlight colors for the target object, set the color attributes in the Highlight Object block to colors from the G2 color set. The color attributes are listed in this table:

The attribute…	Is for region…	Which usually is…
Background Region Color	icon-background	The primary color of the icon.
Outline Region Color	outline	A border around the icon.
Status Region Color	status	An element showing what state the object is in (on or off, for example).
Alarm Region Color	alarm	An element showing that the object has a problem (it is broken, for example).

To prevent the Highlight Object block from changing the color for a particular region, set the color attribute for that region to none.

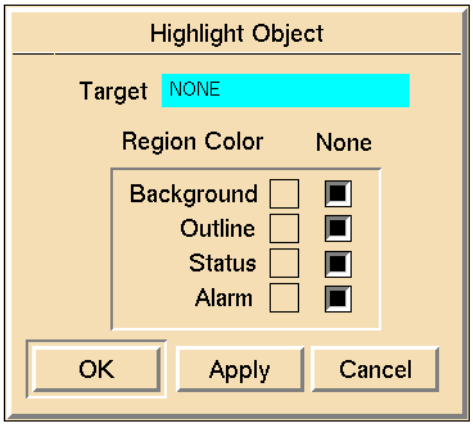
To use the Highlight Object block with an object that has some of but not all four regions, set the color attributes for the missing regions to none. If you specify another value for a missing region, GDA writes a message to the Error Queue.

Resetting

When you reset a Highlight Object block, it changes the colors of the object's regions back to their original colors.

Configuring

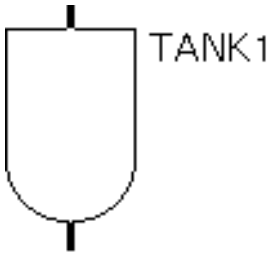
This is the configuration panel for the Highlight Object block.



Attribute	Description
Target	The object whose icon is highlighted.
Background Region Color	The color for the icon-background region.
Outline Region Color	The color for the outline region.
Status Region Color	The color for the status region.
Alarm Region Color	The color for the alarm region.

Example

This example shows how you can use the Highlight Object block to highlight a region or a whole object. It changes the colors of this tank:



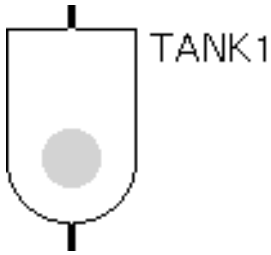
By default the regions have these colors:

The color of the region…	Is...
icon-background	transparent
outline	black
status	transparent
alarm	transparent

The Highlight Object block can highlight just one region. To highlight the status region, set the color attributes to the values shown in this table:

Attribute	Setting
Background Highlight Color	none
Outline Highlight Color	none
Status Highlight Color	light-gray
Alarm Highlight Color	none

The Highlight Object block makes the tank look like this:



The Highlight Object block can also highlight the whole object to make it really stand out. To highlight the tank and make its alarm region visible, set the color attributes to the values shown in this table:

Attribute	Setting
Background Highlight Color	black
Outline Highlight Color	none
Status Highlight Color	black
Alarm Highlight Color	white

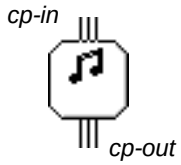
The Highlight Object block makes the tank look like this:

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Sound Bite



The Sound Bite block plays a digitized sound through the workstation's internal speaker. Specify the audio file to play with the attributes Sound File and Sound Directory. Specify the command that plays audio files with the attribute Sound Driver. This block is especially useful when you want to alert an operator with a sound or play a digitized voice message.

This block supports Sun™, Hewlett-Packard™, and Windows workstations.

Note Audio software is not a standard part of the operating system for many workstations. If the audio software is not installed as described in this section, check your workstation's documentation or ask your system administrator for more information.

Playing Sounds on a Sun Workstation

The Sound Bite block is initially set up for Sun workstations, so the attributes Platform, Sound Driver, Sound Directory, and Sound File have default values that work with many Sun workstations. However, you may need to change them depending on how your operating system was installed.

This table lists appropriate values for a Sound Bite block's attribute on a Sun workstation. Note that on a Sun workstation, the Sound Bite ignores the attributes Priority and Loop.

The attribute…	Is the…
Platform	sun
Sound File	File that contains the sound.
Sound Directory	Directory the sound is in. It must end in a slash (/).
Sound Driver	Command that plays sounds. For example, "/usr/demo/SOUND/play" or "/usr/bin/audioplay".

The attribute…	Is the…
Sound Duration	Number of seconds to play the sound.
Volume	Volume to play the sound at. It must be an integer between 1 and 100, where 1 is softest and 100 is loudest.

To pick a value for Sound Driver, look for both commands and use the one you find. In newer versions of the Sun operating system, the command is named audioplay and is in the directory /usr/bin/. In earlier versions, it is named play and is in the directory /usr/demo/SOUND/. If you cannot find the drivers in these two directories, see your System Administrator.

The attribute Duration determines how long Sound Bite waits for the sound to complete. If Duration seconds pass and the sound has not finished, Sound Bite stops the sound and passes the control signal. If the sound finishes before Duration seconds pass, Sound Bite waits until Duration passes before passing the control signal. If Duration is none, Sound Bite always plays the sound until it is finished.

You may find some sample sound files in the directory:

/usr/demo/SOUND/sounds/.

Playing Sounds on a Hewlett-Packard Workstation

For Hewlett-Packard computers, GDA uses the `send_sound` sound driver as its default. This driver handles sound files with these extensions: `.au` and `.wav`. The `SPlayer` sound driver is also supported.

This table lists appropriate values for a Sound Bite block's attributes on a Hewlett-Packard workstation. Note that on a Hewlett-Packard workstation, Sound Bite ignores the Volume attribute.

The attribute...	Is the…
Platform	hp
Sound File	File that contains the sound.
Sound Directory	Directory the sound is in. It must end in a slash (/).
Sound Driver	Command that plays sounds. For example, "/usr/audio/bin/player".
Sound Duration	The number of seconds to play the sound Loop times.

The attribute...	Is the…
Loop	Number of times to play the sound.
Priority	Priority to play the sound at. It can be urgent, hi, normal, or lo.

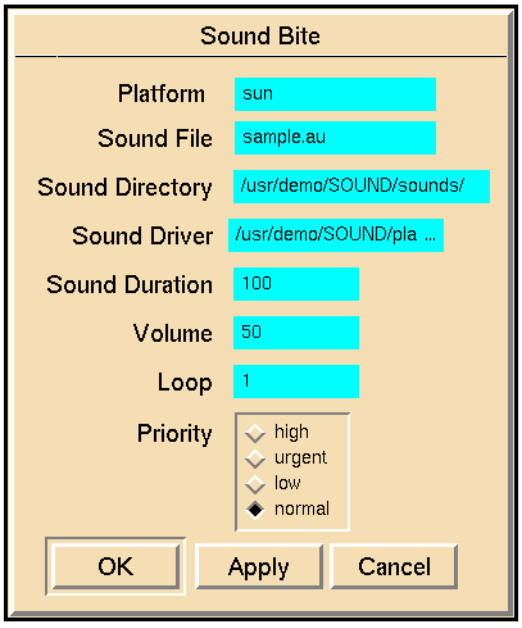
The attribute Duration determines how long Sound Bite waits for the sound to finish playing for the number of times Loop specifies. If Duration seconds pass and the sound has not finished, Sound Bite stops the sound and passes the control signal. If the sound finishes before Duration seconds pass, Sound Bite waits until Duration passes before passing the control signal. To let the sound play until completion, set Duration to none.

Playing Sounds on a Windows Computer

For Windows machines, GDA can use the sound drivers `sndrec32` and `mplay32`, both of which handle `.wav` files. GDA supports the DOS command switches `/PLAY` and `/CLOSE`.

Configuring

This is the configuration panel for the Sound Bite block.



See previous sections for a description of each attribute on the supported platforms.

Example

This table shows how to set up a Sound Bite block on a Sun workstation to play spacemusic.au from /usr/demo/SOUND/sounds/ at a moderate volume, using the audioplay command. It lets the sound play as long as it needs without stopping it.

Set this attribute…	To...
Sound Driver	"/usr/bin/audioplay"
Sound Directory	"/usr/demo/SOUND/sounds/"
Sound File	"spacemusic.au"
Duration	none
Volume	50

This table shows how to set up a Sound Bite block on a Hewlett-Package workstation to play beep.l16 from /usr/audio/examples/ at normal priority, using the player command. It plays the sound two times and waits 30 seconds for the sound to complete twice.

Set this attribute…	To...
Sound Driver	"/usr/audio/bin/player"
Sound Directory	"/usr/audio/examples/"
Sound File	"beep.l16"
Duration	30
Priority	normal
Loop	2

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

17

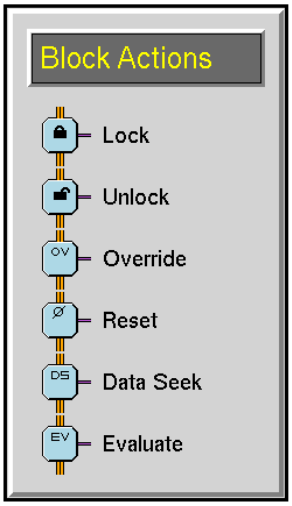
Block Actions

Describes blocks that perform actions on other blocks, using an Action Link.

Introduction

GDA has several blocks that operate on other GDA blocks. These blocks can perform operations similar to the ones in the block's menu, such as locking, unlocking, resetting, and overriding.

You can find the Block Actions palette under the Action submenu of the Palettes menu:



All Block Action blocks also have an Action Link, which you connect to another GDA block. When the Block Action block is evaluated, it performs its action on any block connected to its Action Link. For example, the Lock block locks any block connected to its Action Link.

Performing Menu-Choice Actions on Blocks

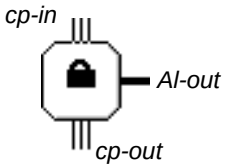
These blocks perform actions that are similar to the menu choices available in a block's menu. Notice that there can be some small but important differences in how these blocks and the menu choices operate.

- The Lock block on page 421 locks a block.
- The Unlock block on page 423 unlocks a locked block. You lock the block with a Lock block. The Unlock cannot unlock a block that the lock menu choice locked.
- The Override block on page 424 overrides the value for a path. It does not ask you for the value with a dialog. Instead, you enter the value in the Override block's attributes.
- The Reset block on page 428 resets a block.
- The Evaluate block on page 431 forces a block to evaluate.

Forcing Data Seeking to Happen

The Data Seek block on page 429 forces data seeking to occur for a variable, which can be an attribute of a GDA block or a stand-alone G2 variable.

Lock



The Lock block locks the blocks connected to its action link. It performs the same action as choosing lock from the blocks' menus. You can override the Lock block with the Override block or the override menu choice. You can unlock the block with the Unlock block or the unlock menu choice.

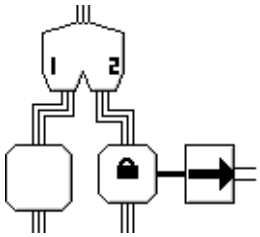
The Lock block is useful when you need to stop a block from passing values.

Configuring

The Lock block has no configurable attributes.

Example

In this figure, a Lock block locks an Entry point, if a user chooses the second option from the User Query Control Switch:



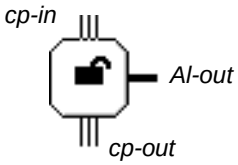
See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Inhibit block	page 138	*Reference Manual*
The Inference Inhibit block	page 303	*Reference Manual*

For more information on...	See...	In this book...
The Control Wait block	page 444	*Reference Manual*
The Control Inhibit block	page 447	*Reference Manual*

Unlock



The Unlock block unlocks the blocks connected to its action link, only if a Lock block locked them. It will not unlock blocks you locked with the lock menu choice. It is similar to the unlock menu choice.

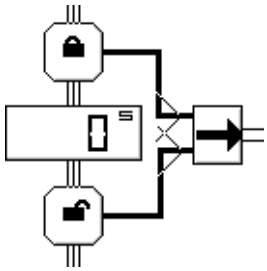
If the block is unlocking an entry point, the entry point is evaluated only if it has received new data since it has been locked. If the block is unlocking any other block, the block is always evaluated.

Configuring

The Unlock block has no configurable attributes.

Example

In this figure, an Entry point is locked and then unlocked after a period of time. A Lock block locks an Entry point, a Control Delay block pauses for ten seconds, and an Unlock block unlocks the Entry Point.

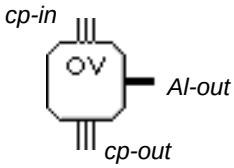


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Override



The Override block overrides the outputs of the inference blocks, Numeric Entry Points, or Control Entry Points connected to its action link. It is similar to choosing the override menu choice, except Override block does not require user input.

GDA determines the output path Quality based on the Override Mode: the Quality is ok when Override Mode is automatic and manual when Override Mode is manual.

Overriding an Inference Block

If the Override block is overriding an inference block or any block with an output inference path, it assigns its attribute values to these inferences path attributes:

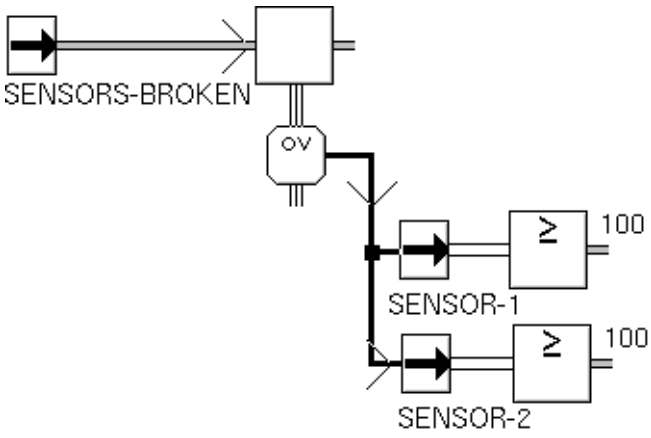
This attribute…	Is the value for the inference path's…
Override Status	Status-value. It is also stored in the blocks ip-out attribute, if the block is an entry point.
Override Belief	Belief-value

Overriding a Numeric Entry Point

If the Override block is overriding a Numeric Entry Point, it assigns its attribute values to this data path attribute:

This attribute…	Is the value for the data path's…
Override Numeric	Data-value. It is also stored in the block's dp-out attribute.

For example, in this figure, an Override block overrides the values of some sensors whenever the sensors are broken:



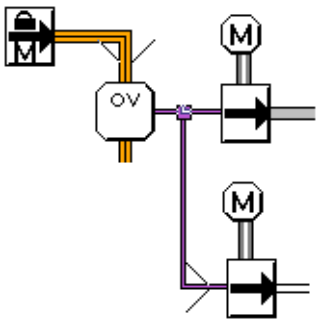
Overriding a Control Entry Point

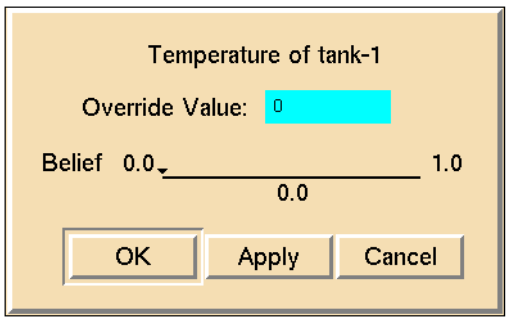
If the Override block is overriding a Control Entry Point, it sends a single control signal to the Control Entry Point.

Overriding Multiple Blocks

You can attach multiple blocks to the Override block's action link. For example, you do this when you want to manually override a number of blocks simultaneously. An example of this occurs in the encapsulation diagram of the User Query blocks as described in "1 Output, 2 Output, 3 Output, 4 Output" on page 469.

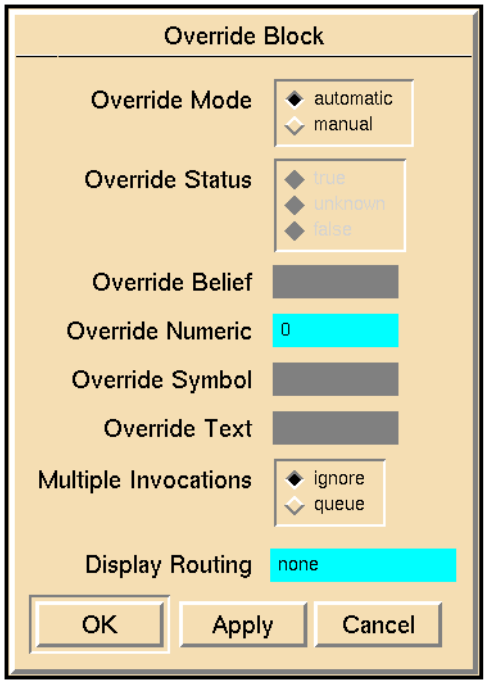
When manually overriding multiple blocks, the override dialog is a composite of the override dialogs associated with each of the attached blocks. The following figure shows the diagram and associated composite dialog that results when sending a control signal to the Override block. The top override is associated with the Numeric Entry Point, and the bottom override is associated with the Belief Entry Point, whose Logic is fuzzy.





Configuring

This is the configuration panel for an Override block that is connected to a Numeric Entry Point. Notice that GDA enables and disables attributes depending on the type of output path whose value(s) is being overridden.



Attribute	Description
Override Mode	Whether to override the output path's attributes automatically (automatic), or whether to force a manual override (manual). Use this feature for blocks that specify the override menu choice.
Override Status	The override Status-value for an output inference path.
Override Belief	The override Belief-value for an output inference path.
Override Numeric	The override Data-value for an output data path of a Numeric Entry Point.
Override Symbol	The override Data-value for an output data path of a Symbolic Entry Point.

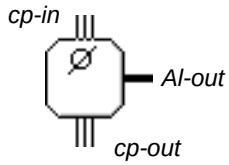
Attribute	Description
Override Text	The override Data-value for an output data path of a Text Entry Point.
Multiple Invocations	See "Specifying How to Handle Multiple Control Signals" on page 438 in the *GDA User's Guide*.
Display Routing	The G2 windows on which the dialog should appear. The options are the name of a display routing object created using the Display Routing block, a G2 window, or none, which displays the dialog in all G2 windows.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Output block	page 145	*Reference Manual*
The Set Attribute block	page 147	*Reference Manual*

Reset



The Reset block resets the blocks connected to its action link. It performs the same action as choosing reset from the blocks' menus.

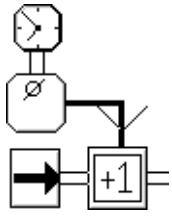
If more than one block is connected to the Reset Blocks's action link, it resets all the blocks.

Configuring

The Reset block has no configurable attributes.

Example

In this figure, a Reset block resets an Input Counter every hour. A Numeric Entry Point is connected to an Input Counter, which passes on the number of times it has received a value. A Reset block with a Clock Capability is also connected to the Input Counter. The Reset block resets the Input Counter back to zero every hour.

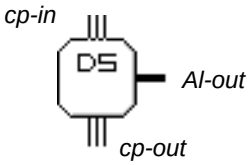


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Data Seek



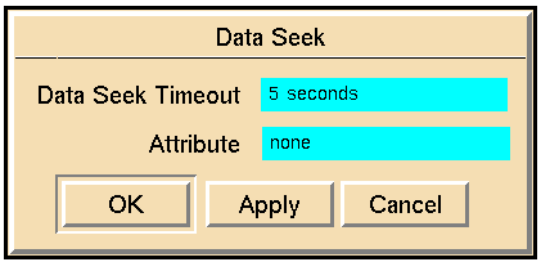
The Data Seek block performs data seeking on a G2 variable, which can be an attribute of a GDA block or a stand-alone G2 variable. For example, you use the Data Seek block to perform data seeking on the dp-out attribute of an entry point. Note that you can also use the Data Seek block to perform data seeking when the dp-out attribute does not refer to a named sensor.

To use a stand-alone G2 variable, connect the Data Seek block's action link to the variable. To use an attribute of a GDA block, connect the Data Seek block's action link to the other block, and set the Attribute attribute to the name of the other block's attribute.

If the block cannot get a value for the variable within the time period in the attribute Data Seek Timeout, the block stops seeking data and signals a GDA error. The value of Data Seek Timeout should be less than the value of the Uninterrupted Procedure Execution Limit, in the Timing Parameters system table. If Data Seek Timeout is none, the block waits for data indefinitely.

Configuring

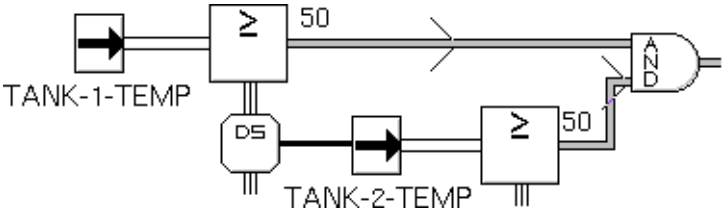
This is the configuration panel for the Data Seek block.



Attribute	Description
Data Seek Timeout	The timeout period for obtaining a value for the variable using data seeking.
Attribute	The attribute of the block to which the action link is attached, which names the variable on which the block performs data seeking.

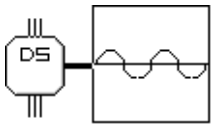
Example

In this figure, a High Observation block checks whether the temperature of Tank 1 is above 50. If it is, the block uses a Data Seek block to find the temperature of Tank 2.



The Data Seek block is connected to its entry point with an action link, and the Attribute attribute of the Data Seek block is dp-out.

In this figure, the Data Seek block is connected to a stand-alone G2 variable. Its Attribute is none.

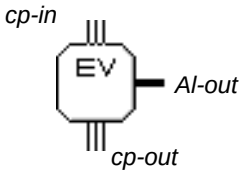


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Output block	page 145	*Reference Manual*
The Set Attribute block	page 147	*Reference Manual*

Evaluate



The Evaluate block evaluates the blocks connected to its action link. It performs the same action as choosing evaluate from the blocks' menus.

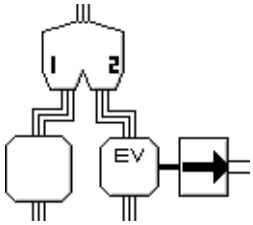
If a connected block has a Clock Capability which has the attribute Allow Intermediate Evaluation set to no, Evaluate block does not evaluate the block. If more than one block is connected to Evaluate Blocks's action link, it evaluates all the blocks.

Configuring

The Evaluate block has no configurable attributes.

Example

In this figure, an Evaluate block evaluates an Entry point, if a user chooses the second option from the User Query Control Switch:

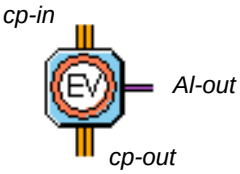


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Enable



The Enable block enables the evaluation of all blocks connected to its action link. If a block or workspace is specified in its Target attribute, the Enable block enables it. The block performs the same action as the enable evaluation menu choice or the gdl-enable-evaluation API procedure, described in the GDA API Reference Manual.

Caution Enabling blocks and workspaces for evaluation is different from enabling blocks and workspaces in G2. For blocks, do not use the enable G2 menu choice. For workspaces, do not use the KB Workspace > Enable G2 menu choice.

Once you have enabled a block or workspace by using the Enable block, the specified block or workspace remains enabled until you disable evaluation using the Disable block, the disable evaluation menu choice, or the gdl-disable-evaluation API procedure.

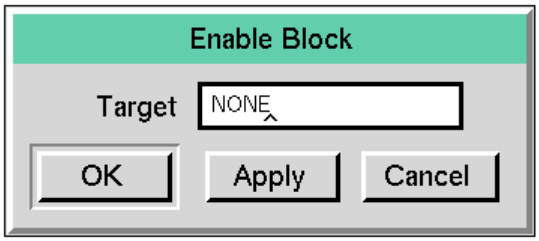
The Target attribute specifies the name of the block or workspace to be enabled. The Enable block posts an error if the Target attribute does not refer to a valid block or workspace.

To specify a block or workspace to be enabled:

1. Clone the Enable block in the Palettes > Block Actions palette to your workspace.
2. Connect the Enable block to the specified block using an action link, or select Configure from the table for the specified Enable block and go to the next step.
3. Type in the name of the block or workspace in the Target field of the Configure dialog, shown below.

Configuring

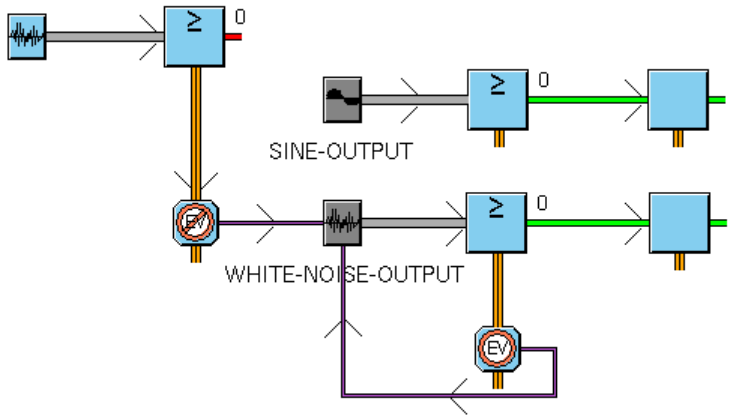
This is the configuration panel for the Enable block.



Attribute	Description
Target	The name of the block or workspace to be enabled.

Example

The following figure shows a diagram with an Enable block whose Target attribute has the value Sine-output. Its action link is connected to a White-noise-output block. When the Enable block receives a control signal it enables evaluation in the Sine-output and White-noise-output blocks, which were disabled by the Disable block upstream of the Enable block.

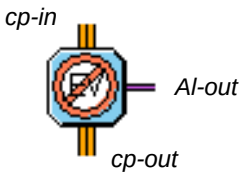


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Disable



The Disable block disables the evaluation of all blocks connected to its action link. If a block or workspace is specified in its Target attribute, the Disable block disables it. The block performs the same action as choosing the disable evaluation menu item or the gdl-disable-evaluation API, described in the GDA API Reference Manual.

Caution Disabling blocks and workspaces for evaluation is different from disabling blocks and workspaces in G2. For blocks, do not use the disable G2 menu choice. For workspaces, do not use the KB Workspace > Disable G2 menu choice.

When disabled, the specified block becomes gray and GDA performs no evaluation on it.

Once you have disabled a block or workspace by using the Disable block, the specified block or workspace remains disabled until you enable evaluation using the Enable block, the enable evaluation menu item, or the gdl-enable-evaluation API procedure.

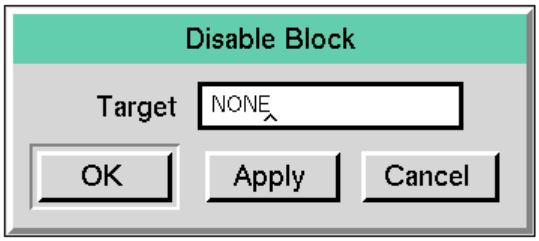
The Target attribute specifies the name of the block or workspace to be disabledl The Disable block posts an error if the Target attribute does not refer to a valid block or workspace.

To specify a block or workspace to be enabled:

1. Clone the Disable block in the Palettes > Block Actions palette to your workspace.
2. Connect the Disable block to the specified block using an action link, or select Configure from the table for the specified Disable block.
3. Type in the name of the block or workspace in the Target field of the Configure dialog, shown below.

Configuring

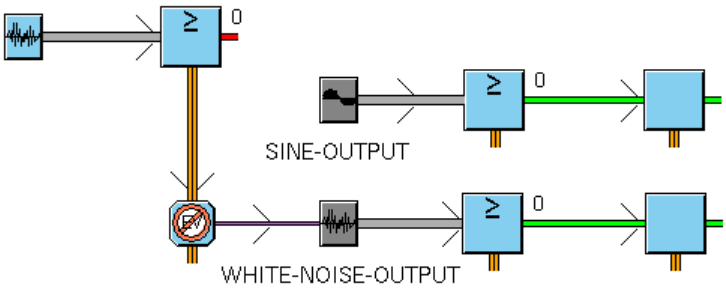
This is the configuration panel for the Disable block.



Attribute	Description
Target	The name of the block or workspace to be disabled.

Example

The following figure shows a Disable block whose Target attribute has the value Sine-output. Its action link is connected to a White-noise-output block. When the Disable block receives a control signal, it disables the Sine-output block and White-noise-output block.



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

18

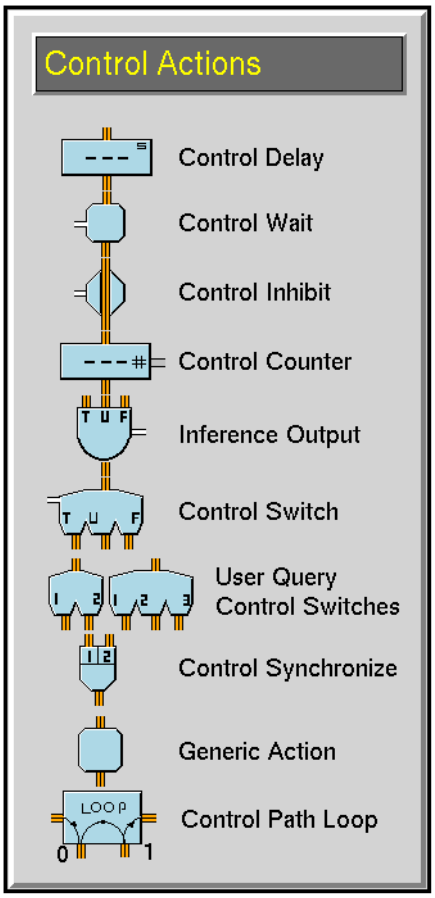
Control Actions

Describes blocks that determine how control signals flow through a diagram.

Introduction

These blocks let you determine how control signals flow through your diagram. You can stop a control signal, divert it, branch to different paths, and evaluate your own G2 procedures.

You can find the Control Actions palette in the Action submenu of the Palettes menu:



The Control Actions also let you create loops that execute actions repeatedly. To see a loop that executes a specified number of times, see the example under “Control Counter” on page 449. To see a loop that executes until some condition is met, see the example under “Control Switch” on page 454.

Specifying How to Handle Multiple Control Signals

Some of the blocks on the Control Actions palette let you choose what the block does when it receives a control signal while it is already executing. In these blocks, the attribute Multiple Invocations specifies how to handle multiple control signals.

Generally, the blocks with the attribute Multiple Invocations are those that may take a significant amount of time to evaluate. On the Control Actions palette, those blocks are Control Delay, Control Wait, Control Counter, User Query Control Switches, and Generic Action.

Here are the default settings for the Multiple Invocations attribute for these blocks:

This block...	Has this default...	And options...
Control Delay	ignore	ok
Control Wait	ignore	queue ok
Control Counter	ignore	ok
User Query Control Switches	ignore	queue ok
Generic Action	ignore	queue ok

For information on what each option means, see "Specifying How to Handle Multiple Values" on page 92 in the *GDA User's Guide*.

Stopping and Pausing Paths

These blocks let you stop the flow of control signals or pause it until some condition is met:

- The Control Delay block on page 441 pauses the flow of a control signal for a set period of time.
- The Control Wait block on page 444 holds incoming control signals until an inference input has a given value.
- The Control Inhibit block on page 447 discards an incoming control signal if an inference input has a given value.
- The Control Synchronize block on page 461 synchronizes two control paths by waiting for a signal from both paths to arrive before passing a signal.
- The Control Path Loop block on page 465 allows you to add control loops to a diagram, which terminate based on a termination limit or based on an expression turning true.

Branching

These blocks let you choose the control path on which to send a control signal:

- The Control Switch block on page 454 lets an inference path choose the path on which to send a control signal.

- The User Query Control Switches block on page 456 let the operator choose which paths to send control signals down. They display a dialog box that lets the operator choose one or more paths.

Outputting Data

These blocks let you pass an inference value or a count of control signals.

- The Control Counter block on page 449 passes on a data path the number of times it received a control signal. It also passes the control signal on an output control path.
- The Inference Output block on page 451 chooses which inference value to pass depending on which of its three input control paths on which it receives a signal.

Defining Your Own Action

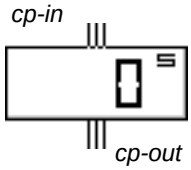
The Generic Action block on page 463 can execute a G2 procedure you have written when it receives a control signal.

See Also

GDA has many other blocks that control the flow of information, but that take data values or inference values as input:

- To control the flow of Data values, use the blocks on the Data Control palette in the Data Blocks menu.
- To control the flow of inference values, use the blocks on the Logic Gates palette and the Temporal Gates palette in the Inference Blocks menu.
- To apply your own function to an input data value, use the Arithmetic Function block on page 122 on the Functions palette.

Control Delay



The Control Delay block delays passing its control signal for a specified amount of time, and displays a countdown of the time on its icon. To specify the length of the delay, set the Delay attribute. To specify how the icon displays the countdown, set the attributes Count By and Display Units.

This block is useful for simulating system delays, such as:

- Material handling, such as moving a part from one area of a factory to another.
- Mechanical delays, such as waiting for a valve to move to its fully open position.

Specifying How to Display the Countdown

To specify how often the icon updates the countdown, set the Count By attribute. To specify the unit of time it displays, set the Display Units attribute to seconds, minutes, hours, or days. This figure shows how the Control Delay displays 30 seconds, 45 minutes, 72 hours, and 7 days:

The icon can display numbers up to 999. If the block cannot display the time interval in the units you specified, it tries the larger units in order until it can. For example, if you set Delay to 2700 seconds and Display Units to seconds, the block displays "45 m."

If the block needs to round the time it displays, it rounds to the nearest whole number, rounding up when it is halfway between whole numbers. For example, it displays 1 $\frac{1}{4}$ hours as "1 h," 1 $\frac{3}{4}$ hours as "2 h," and 1 $\frac{1}{2}$ hours as "2 h."

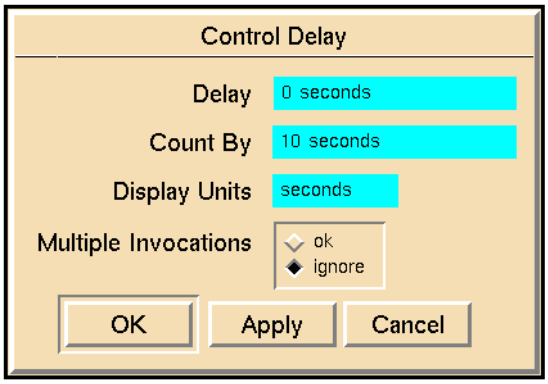
Note When Multiple Invocations is set to ok, every invocation results in a separate delay cycle. In this case, the countdown might not appear correct because the block is showing multiple cycles on the same display.

Resetting

When you reset this block, it aborts any outstanding delays and passes on its signals immediately.

Configuring

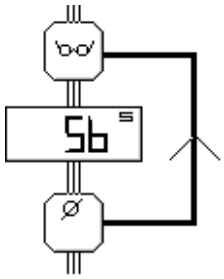
This is the configuration panel for the Control Delay block.



Attribute	Description
Delay	The amount of time that the counter delays passing its input.
Count By	The frequency with which the counter updates the countdown.
Display Units	The unit of time that the counter displays.
Multiple Invocations	See “Specifying How to Handle Multiple Control Signals” on page 438.

Example

The diagram in this figure shows the subworkspace of the Show Subworkspace block, pauses for 1 minute, and then hides the subworkspace. The Control Delay block is counting down from 1 minute.

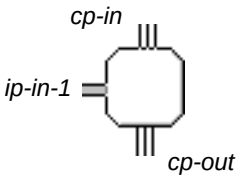


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Handle Multiple Control Signals	page 438	*Reference Manual*
The Data Delay block	page 130	*Reference Manual*
The Inference Delay block	page 363	*Reference Manual*

Control Wait



The Control Wait block lets you pause a control signal until an inference path has a certain value. The attribute Trigger On determines which inference values let the control signal pass. The block passes all pending control signals when the value of the block's input inference path matches Trigger On.

Note The Control Inhibit block is similar, but it discards its control signal instead of pausing it. The attribute Trigger On specifies which inference value discards the control signal, not which value lets it pass.

Handling Multiple Control Signals

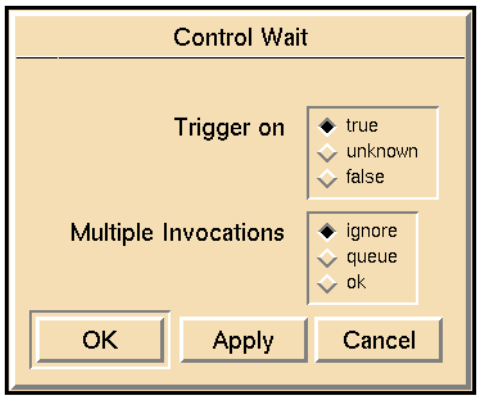
The following table summarizes how the block stores and passes control signals based on the value of the Multiple Invocations attribute:

If Multiple Invocations is...	The block stores the following while waiting for the Trigger On symbol...	And the block passes the following when it receives the Trigger On symbol...
ignore (the default)	one control signal	the current control signal
queue	multiple control signals	one control signal at a time
OK	multiple control signals	all control signals

When this block is locked, it discards its input control signal.

Configuring

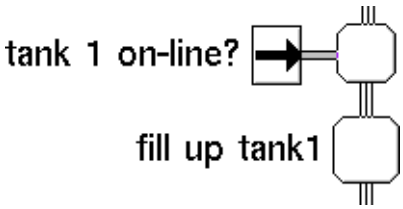
This is the configuration panel for the Control Wait block.



Attribute	Description
Trigger On	The truth value that allows the control signal to pass.
Multiple Invocations	See “Specifying How to Handle Multiple Control Signals” on page 438.

Example

The diagram in the following figure waits for tank 1 to be on-line before it fills the tank up. If the Control Wait block receives a control signal before the tank is online, the block holds it until the tank is online and then passes it.

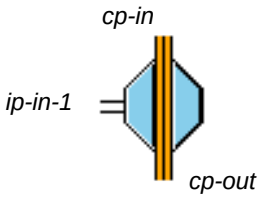


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Handle Multiple Control Signals	page 438	*Reference Manual*
The Data Delay block	page 130	*Reference Manual*
The Inference Delay block	page 363	*Reference Manual*

Control Inhibit



The Control Inhibit block lets an inference path turn a control path on and off. You can use the block to turn on and off entire sections of a flow diagram.

When the status value of the block's input inference path matches the value of the attribute Trigger On, the block discards the input control signal. When the status value of the inference path no longer matches Trigger On, the block passes the current value of the input control path and continues to pass the input control signal normally.

If the block's inference path hasn't received a value yet (that is, it has a quality of no-value), the block passes nothing even when it receives a value from its control path.

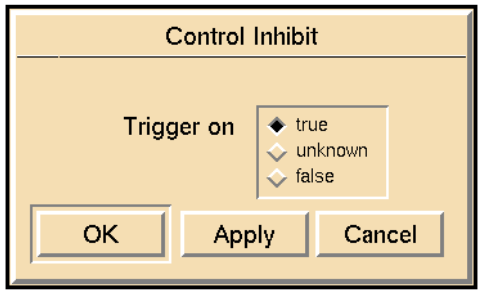
Note The Control Wait block is similar, but it pauses its control signal instead of discarding it. The attribute Trigger On specifies which inference value lets the control signal pass, not which value stops it.

Resetting

When you reset a Data Inhibit Block, the block does not does not pass a signal until it receives a value from its inference input path, even if it receives a signal from its input control path.

Configuring

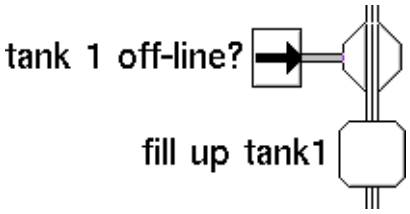
This is the configuration panel for the Control Inhibit block.



Attribute	Description
Trigger On	The truth value that causes the block to discard its input control signal.

Example

The diagram in this figure fills up tank 1 only if the tank is online. If the Control Inhibit block receives a control signal while the tank is off-line, the block discards the signal. It must receive a signal while the tank is online before it will fill up the tank.

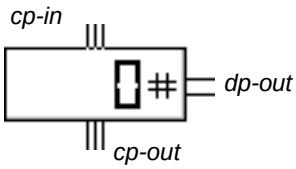


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Handle Multiple Control Signals	page 438	*Reference Manual*
The Data Delay block	page 130	*Reference Manual*
The Inference Delay block	page 363	*Reference Manual*

Control Counter



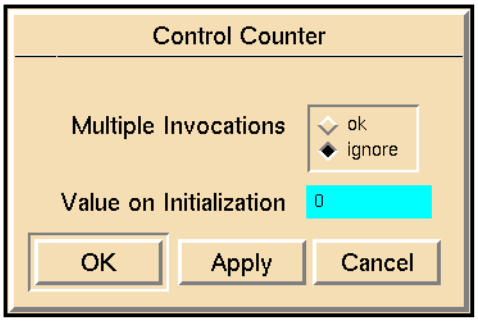
The Control Counter counts the number of times it receives a control signal and displays the count on its icon. It passes the control signal on its output control path and the count on its output data path.

Resetting

When you reset this block, it resets its counter to the value of the attribute Value on Initialization and passes that value on its data path.

Configuring

This is the configuration panel for the Control Counter.

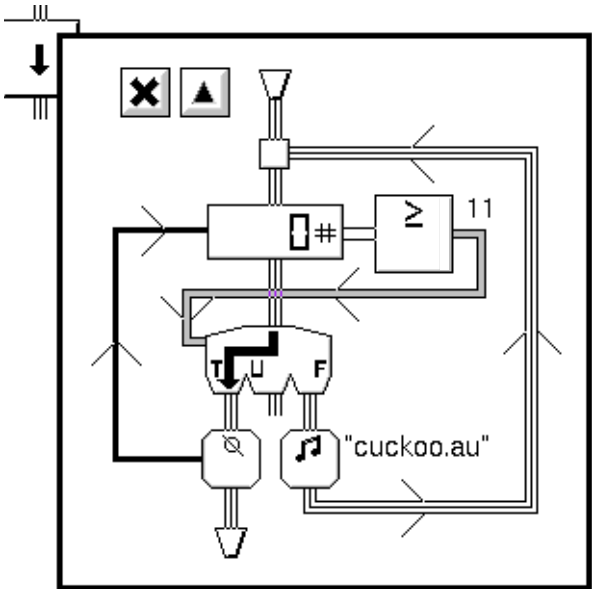


Attribute	Description
Multiple Invocations	See "Specifying How to Handle Multiple Values" on page 92 in the *GDA User's Guide*.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide* .

Example

The diagram in this figure plays the sound file cuckoo.au ten times. The Control Counter counts the number of times cuckoo.au has played. The High Value block then checks if the sound has been played ten times. If the High Value block passes

.false, the Control Switch passes control to the Sound Block which plays cuckoo.au and passes control back to the Control Counter. If the High Value block passes .true, the Control Switch passes control to the Reset Block which resets the Control Counter and stops.



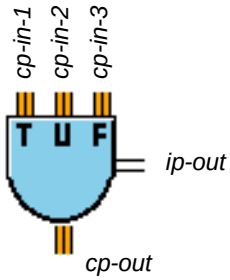
To start this diagram, choose evaluate from the Control Counter's menu.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying an Initial Data Value	page 83	*User's Guide*
The Input Counter block	page 150	*Reference Manual*
The Inference Counter block	page 372	*Reference Manual*

Inference Output



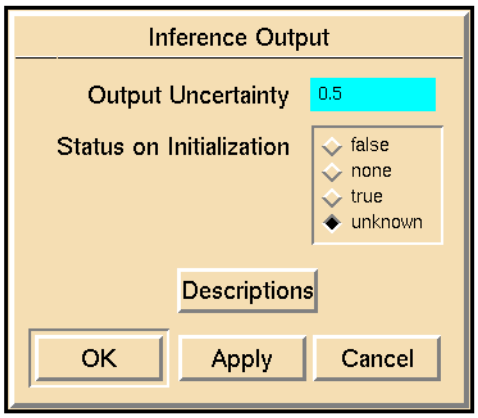
The Inference Output block chooses an inference value to pass depending on which input control path it receives a signal from. Whenever it receives a control signal on one of its input control paths, it passes the inference value for that path on its output inference path and passes the control signal on its output control path.

This table lists the inference values associated with each input control path:

If the block receives a control signal on the path marked…	It passes this inference value…
"T" (cp-in-1)	.true
"U" (cp-in-2)	unknown
"F" (cp-in-3)	.false

Configuring

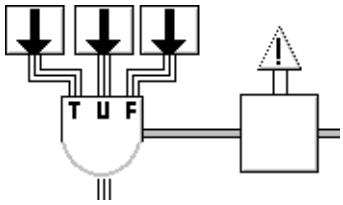
This is the configuration panel for the Inference Output block.



Attribute	Description
Output Uncertainty	See “Specifying Uncertainty” on page 102 in the *GDA User’s Guide*.
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide* .
Description when True, Description when False, Description when Unknown, and Description of Input	See “Specifying an Explanation” on page 94 in the *GDA User’s Guide*.

Example

The diagram in this figure raises an alarm if the left-most Control Entry Point passes a control signal:



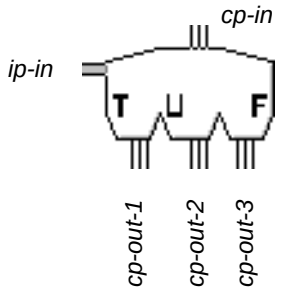
See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User’s Guide*
Specifying and Generating Explanations	page 93	*User’s Guide*
Specifying Uncertainty	page 102	*User’s Guide*
Specifying an Initial Status Value	page 84	*User’s Guide*
The Data Output block	page 145	*Reference Manual*
The Belief Input block	page 253	*Reference Manual*

For more information on...	See...	In this book...
The Belief Output block	page 340	*Reference Manual*
The User Query blocks	page 469	*Reference Manual*

Control Switch



The Control Switch lets an inference value choose which path to send a control signal down. It is useful when you need to perform a sequence of control blocks until some condition is met.

The Control Switch block only passes values when it receives a new control signal; it does not pass values when the inference value changes.

The value of the side inference path (ip-in) determines where the control signal (cp-in) will go. This table shows on which output path the signal will go:

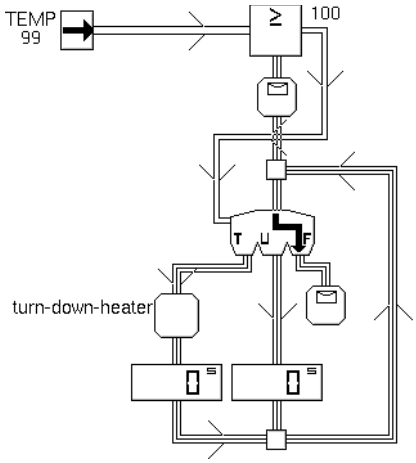
If the side input is…	The left input is passed on the port labeled…
.true	"T" (cp-out-1)
unknown	"U" (cp-out-2)
.false	"F" (cp-out-3)

Configuring

The Control Switch has no configurable attributes.

Example

The diagram in this figure turns down a tank's heater until the tank is no longer overheated. If the temperature is over 100, GDA passes a control signal down the High Value block's control path and sends a message to the Message Queue. It then enters a loop which tries to turn down the tank's heater until the temperature is below 100.



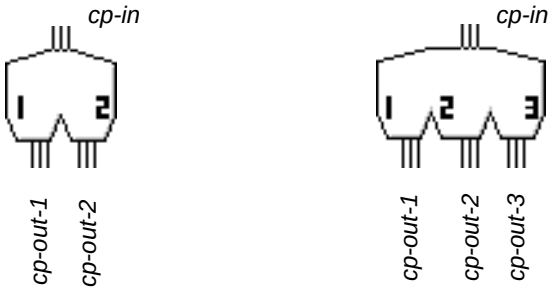
The Control Switch uses the inference value from the High Value block to determine what to do. If High Value passes .true, the diagram calls the procedure turn-down-heater, delays the control signal for a few seconds to let the change take effect, and then passes the signal back to the Control Switch. If High Value passes unknown, the diagram delays the control signal for a few seconds to let the entry point get a new value and then passes the signal back to the Control Switch. If High Value passes .false, the diagram sends a message to the Message Queue saying that the tank's temperature is OK.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Switch block	page 140	*Reference Manual*
The Inference Switch block	page 305	*Reference Manual*
The User Query Control Switches	page 456	*Reference Manual*

User Query Control Switches



A User Query Control Switch lets you choose which control paths receive a control signal. It displays a dialog that lists radio buttons for two or three control paths, and it sends control signals down the paths you specify. The Two-Option User Query Switch lets you choose between two paths, and the Three-Option User Query Switch lets you choose between three paths.

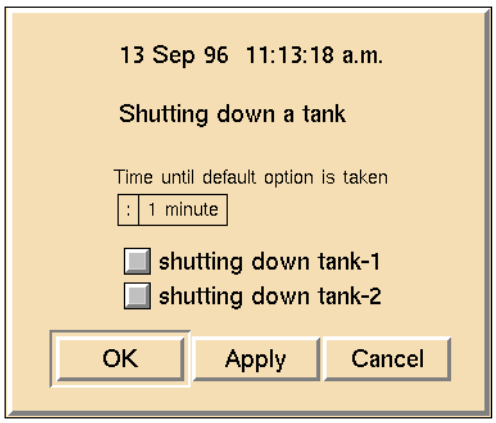
To put a message of your own in the dialog, use the attribute Message Text. To customize the descriptions of the radio buttons, use the attributes Option 1 Description, Option 2 Description, and Option 3 Description.

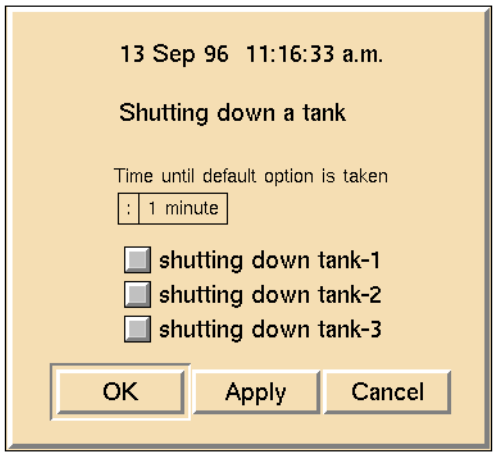
Specifying the Contents of the Dialog

The dialogs are shown in the following figure. The dialog contains the time the dialog appeared and the contents of the attribute Message Text. If you do not specify a value for Message Text, the dialog contains just the time stamp.

Under the two dialogs is a warning that tells you how much longer you have to respond before the block uses the default choice. The block updates the time in this message every 15 seconds.

Next are the radio buttons you press to select control paths. You can specify descriptions for the paths with the attributes Option 1 Description, Option 2 Description, and Option 3 Description. If you do not specify a value for these descriptions, the block displays "activate output 1", "activate output 2", and "activate output 3".





Note The attributes Option 1 Description, Option 2 Description, and Option 3 Description can include the values of certain G2 expressions, as described in "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*. For example, the string "Shut down the [the contents of tank-1] tank" might appear as "Shut down the water tank", where water is the contents of tank-1.

At the bottom are the buttons OK and Cancel. If you press OK, the block passes a signal on the paths you chose, and the dialog disappears. If you press Cancel, the block passes nothing, and the dialog disappears.

Specifying a Timeout Period

To specify what happens when you do not make a selection within a certain amount of time, use the attributes Timeout and Default Option. Timeout is the amount of time you have to make a selection. If Timeout passes and you have not selected OK or Apply, the block takes the action specified in Default Option. If Default Option is 1, 2, or 3, the block passes the control signal down that path. If Default Option is 0, the block discards the control signal and passes nothing.

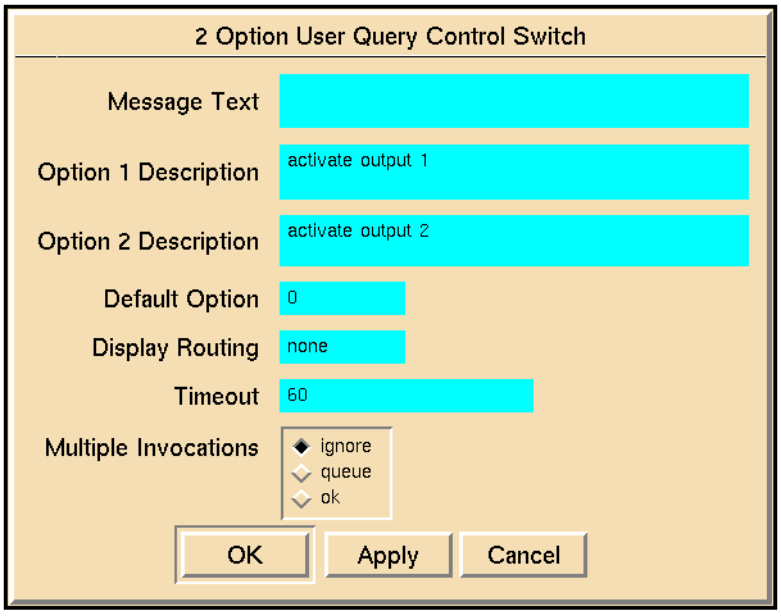
Specifying in which G2 Windows the Dialog Box Appears

To specify in which G2 windows the dialog should appear, use the Display Routing attribute. It can have the values in this table:

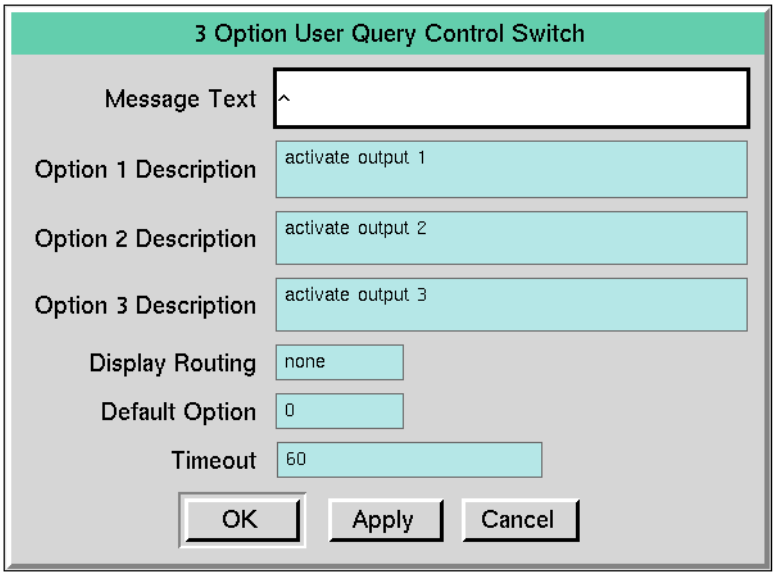
If Display Routing is...	The dialog box appears in these G2 windows...
A Display Routing object	All the windows specified in the object
A G2 Window	That G2 window
none	All G2 windows

Configuring

This is the configuration panel for the Two-Option User Query Switch.



This is the configuration panel for the Three-Option User Query Switch.



Attribute	Description
Message Text	A message to display in the dialog box that appears.
Option 1 Description	The text that appears next to the option that sends the control signal onto the first control output path (cp-out-1).
Option 2 Description	The text that appears next to the option that sends the control signal onto the second control output path (cp-out-2).
Option 3 Description	For a 3 Option User Query Control Switch, the text that appears next to the option that sends the control signal onto the third control output path (cp-out-3).
Display Routing	The G2 window in which the dialog box appears.
Default Option	The default path onto which the control signal is sent when the Timeout period has elapsed. The options are 1, 2, 3, or 0, to discard the input signal.

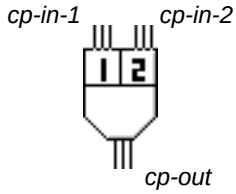
Attribute	Description
Timeout	The time period in which the user has to choose an option before the block uses the Default Option.
Multiple Invocations	See "Specifying How to Handle Multiple Control Signals" on page 438.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
Specifying an Initial Status Value	page 84	*User's Guide*
Specifying How to Handle Multiple Control Signals	page 438	*Reference Manual*
The Data Switch block	page 140	*Reference Manual*
The Inference Switch block	page 305	*Reference Manual*
The Control Switch	page 454	*Reference Manual*

Control Synchronize



The Control Synchronize block synchronizes two control paths by passing a control signal only after it receives a signal on both its input control paths. In effect, it discards the first control signal it receives.

Resetting

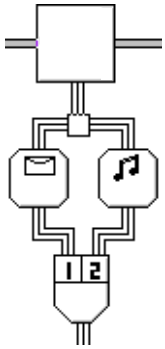
When you reset a Data Synchronize block, it discards any pending input control signal. Also, resetting a Control Synchronize block does not cause a control token to be propagated. This is to prevent triggering of a chained synchronize block by a single input if the blocks are reset in reverse order (downstream to upstream).

Configuring

The Control Synchronize block has no configurable attributes.

Example

The Observation block in this figure sends a control signal to both a Queue Message block and a Sound Bite block. The Control Synchronize block passes one control signal when the two other blocks have both completed.

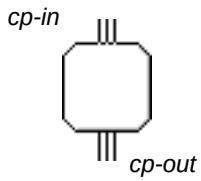


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Data Synchronize block	page 136	*Reference Manual*
The True if Expired block	page 297	*Reference Manual*

Generic Action



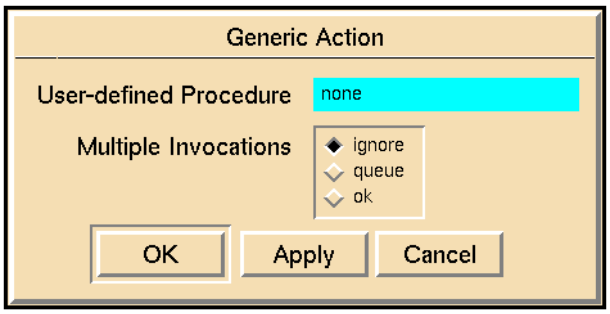
The Generic Action block lets you run a G2 procedure you have written. Set the attribute User Defined Procedure to the name of the procedure.

Define the procedure as shown below. Notice that it takes one argument, the Generic Action block that is calling it.

```
generic-evaluator (actionBlock: class gdl-generic-action)
begin
    {Your code goes here}
end
```

Configuring

This is the configuration panel for the Generic Action block.

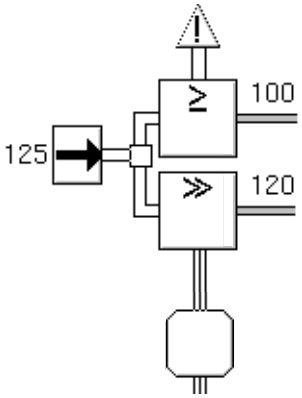


Attribute	Description
User Defined Procedure	The name of the user-defined procedure that the block executes.
Multiple Invocations	See "Specifying How to Handle Multiple Control Signals" on page 438.

Example

The Generic Action block in this figure sends a mail message to a manager, warning that Tank 1 is dangerously overheating. The attribute User Defined

Procedure is set to mail-warning, which is listed below. Notice that when the temperature is over 100, the diagram raises an alarm, and when the temperature is over 120, it raises an alarm and sends mail to the manager.



This is how the G2 procedure mail-warning is defined:

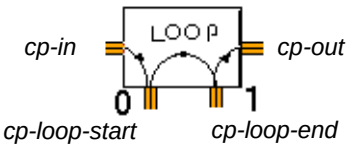
```
mail warning (block: class gdl-generic-action)

file-name: text = "/tmp/mail-file.txt";
tmp-file: class g2-stream;

begin
    tmp-file = call g2-open-file-for-write(file-name);
    call g2-write-line(tmp-file, "WARNING!");
    call g2-write-line(tmp-file, "Tank is dangerously overheating.");
    call g2-write-line(tmp-file, "It may need to be replaced.");
    call g2-close-file(tmp-file);
    call g2-spawn-process-to-run-command-line
        ("mail manager < [file-name]");
end
```

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying How to Handle Multiple Control Signals	page 438	*Reference Manual*
The Arithmetic Function block	page 122	*Reference Manual*

Control Path Loop



The Control Path Loop block allows you to add control loops to a diagram. This block iterates from 0 to the value specified by the Iteration Limit attribute, or until the text expression in the Exit If attribute turns .true.

For more information on how to specify text expressions, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.

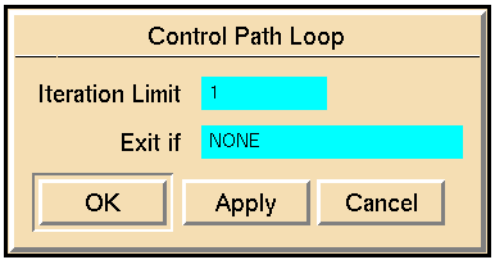
Diagrams that include the Control Path Loop block must contain a circuit breaker somewhere in the cycle.

Resetting

When you reset the Control Path Loop, the block sets its counter back to zero.

Configuring

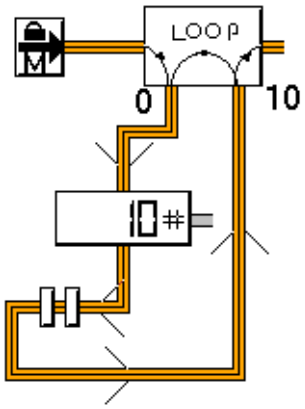
This is the configuration panel for the Control Path Loop block.



Attribute	Description
Iteration Limit	The number of times that the block executes.
Exit If	An expression that when .true, causes the block to stop executing. For information on specifying the expression, see "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*.

Example

The following example illustrates how you use this block to create a loop that counts from 0 to 10, inclusive:



For more information on...	See...	In this book...
Resetting Blocks	page 50	*User's Guide*
Evaluating Blocks	page 51	*User's Guide*
Reading Notes and Errors	page 48	*User's Guide*

19

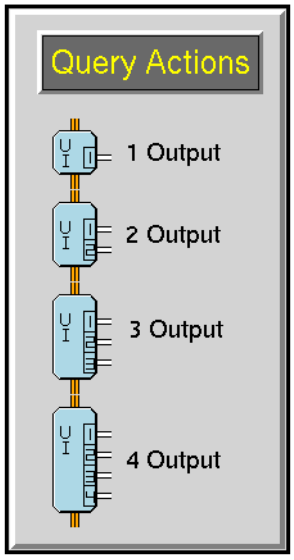
Query Actions

Describes the blocks that let you choose the values for inference paths.

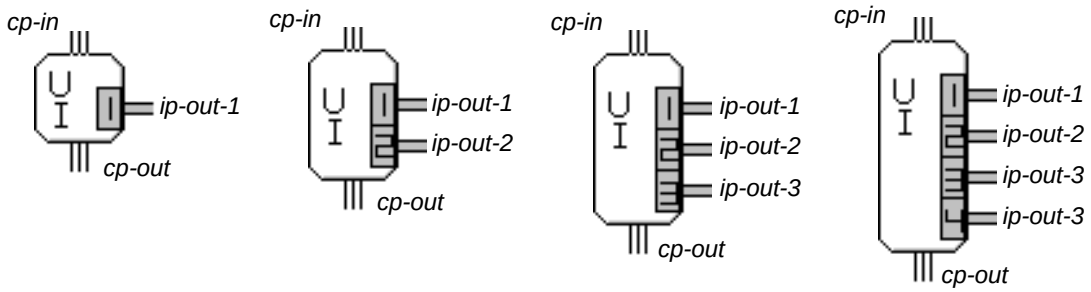
Introduction

The blocks in the Query Actions palette let you choose the inference values for 1 to 4 inference paths. They display a dialog that lists three choices (.true, unknown, or .false) for each inference path.

You can find the Query Actions palette in the Action submenu of the Palettes menu:

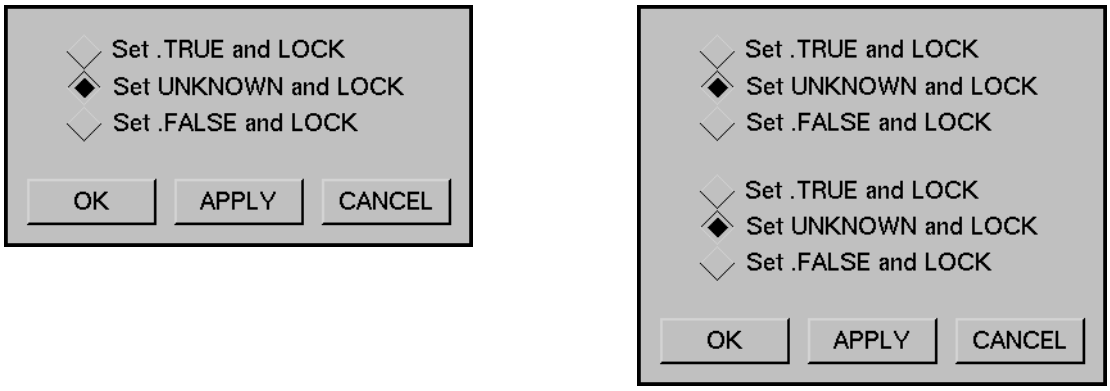


1 Output, 2 Output, 3 Output, 4 Output

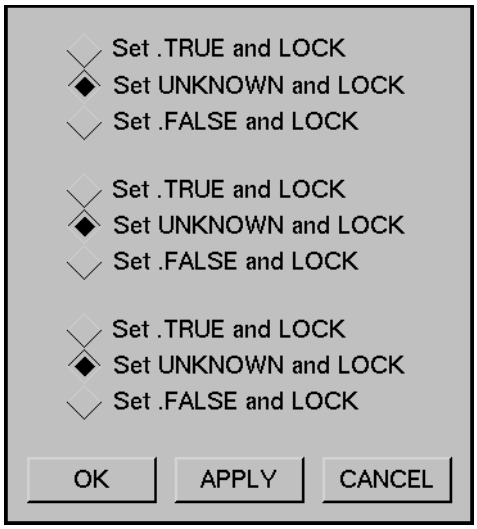


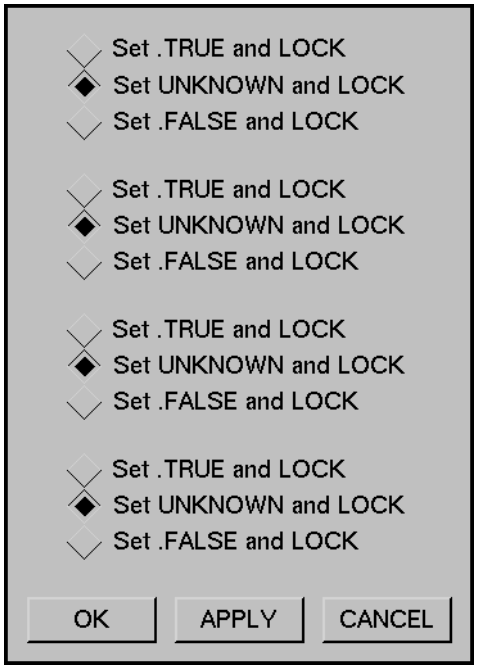
A User Query block lets you choose the output value for 1 to 4 inference paths. The block displays a dialog with three radio buttons for each path: .true, unknown, and .false. The dialog looks like several override dialogs combined together.

This figure shows the default dialogs for One-Output and Two-Output User Query blocks.



The following figure shows the default dialogs for Three-Output and Four-Output User Query blocks.



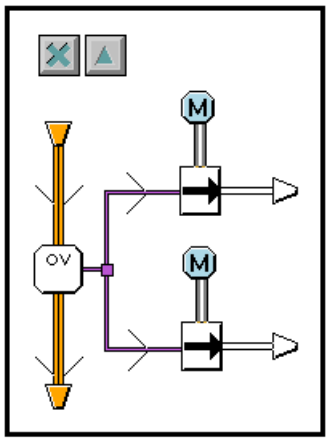


If you press OK, the block passes the value you choose, and the dialog disappears. If you press Apply, the block passes the value you choose, and the dialog remains on the screen. If you press Cancel, the block passes nothing, and the dialog disappears.

Viewing the Diagram of a User Query Block

User Query blocks are encapsulation blocks. The diagram associated with each type of block contains an Override block with an input and output control path, whose action links are connected to Belief Entry Points with Manual Entry Restrictions. The number of Belief Entry Points corresponds to the number of output inference paths on the User Query block.

To view the diagram associated with the encapsulation, select view diagram from the User Query Block's menu. The figure below shows the diagram associated with the 2 Output User Query block:



The Override Mode attribute for the Override block is set to manual by default to enable manually overriding the attached blocks.

Note For information on how to convert User Query blocks created using GDA 1.1 to GDA 2.0, see "Upgrading Your KB to GDA 2.0" in the *GDA 2.0 Installation Guide*.

Specifying How the Block Handles Multiple Control Signals

To specify how the block handles multiple control signals, configure the Multiple Invocations attribute of the Override block in the User Query block's diagram. The options are ignore and queue, as described in "Specifying How to Handle Multiple Values" on page 92 in the *GDA User's Guide*.

Creating a User Query Block That Passes Data Values

To create a user query block that passes data as opposed to inference values, create an encapsulation block with an input and output control path and one or more output data paths. Configure the diagram to look like the subworkspace of a User Query block, except use Numeric Entry Point blocks instead of Belief Entry Point blocks.

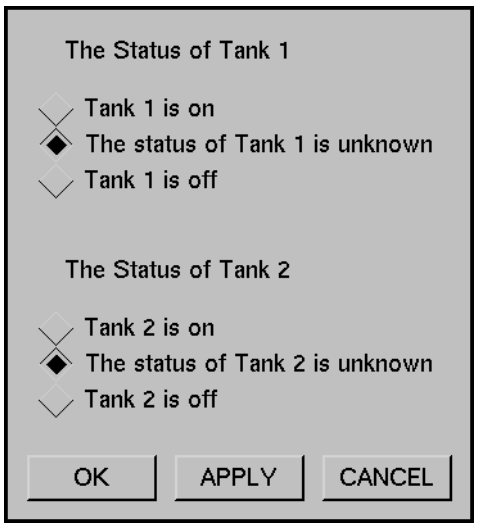
Specifying the Dialog's Text

You can use either the default wording for the dialog, shown in the dialogs at the beginning of this section, or you can use your own wording. To specify your own wording, you need to configure the blocks on the User Query block's subworkspace, as described below:

1. Choose view diagram from the User Query block's menu.
2. For each entry point on the subworkspace, edit the attributes Description When True, Description When Unknown, and Description When False.
3. To add a description over an entry point's radio buttons, change the Override Text attribute in the Manual Restriction attached to the entry point.

For example, this table shows the attribute settings that produced the dialog in the following figure:

This attribute…	Of this block…	Is set to this value…
Override Text	First Manual Restriction	"The Status of Tank 1:"
Description When True	First Entry Point	"Tank 1 is on"
Description When Unknown	First Entry Point	"The status of Tank 1 is unknown"
Description When False	First Entry Point	"Tank 1 is off"
Override Text	Second Manual Restriction	"The Status of Tank 2:"
Description When True	Second Entry Point	"Tank 2 is on"
Description When True	Second Entry Point	"The status of Tank 2 is unknown"
Description When True	Second Entry Point	"Tank 2 is off"



Specifying the G2 Windows in which the Dialog Appears

To specify the G2 windows in which the dialog should appear, specify the Display Routing attribute of the Override block on the subworkspace of the Query Action block. For more information, see "Override" on page 424.

Note Display Routing is an object on the Network palette in the Other Tools menu. It lets you give a name to a group of G2 windows. A G2 window can be a Telewindows connection to a G2 process or a G2 process.

Configuring

User Query blocks are encapsulation blocks; they have no configuration panels of their own. To configure a User Query block, choose view diagram from the block's menu, then configure the blocks on the subworkspace.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Belief Input block	page 253	*Reference Manual*
The Belief Output block	page 340	*Reference Manual*
The Inference Query block	page 381	*Reference Manual*
The Inference Output blocks	page 451	*Reference Manual*

Part IV

Other

20

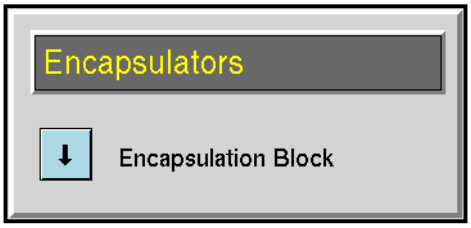
Encapsulation

Describes how to create blocks with subworkspaces that contain GDA diagrams.

Introduction

The block on the Encapsulators palette lets you hide complexity in a GDA diagram by placing a portion of the diagram on a subworkspace.

You find the Encapsulators palette in the Other submenu of the Palettes menu:



You use traditional encapsulation to create single instances of an encapsulation block when the definition of the encapsulation does not need to be propagated to other blocks.

You can also use the GDA wizard to create subclasses of Encapsulation blocks and Single Source Encapsulation (SSE) blocks, which you can place on a palette and clone as needed in a diagram. Single source encapsulation is a powerful tool for maintaining consistency among cloned diagrams encapsulated on a subworkspace.

For more information about creating subclasses of SSEs, see "Single Source Encapsulation" on page 208 in the *GDA User's Guide*.

For general information about using the wizard to create custom subclasses, see "Custom Block Wizard" on page 169 in the *GDA User's Guide*.

Note While it is possible to reuse traditional Encapsulation blocks, the use of Single Source Encapsulation blocks is the preferred way of creating reusable diagrams.

Encapsulation Block

The Encapsulation block lets you place a portion of a GDA diagram in a subworkspace, which is useful for managing complexity in a diagram.

Note If you need to repeat a particular pattern of blocks in more than one place and propagate changes to other blocks of the same subclass, use a Single Source Encapsulation block on page 208 in the *GDA User's Guide*.

The following headings describe how to create the stubs and specify the details of the encapsulation.

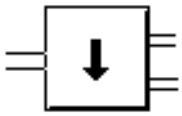
Creating the Stubs for the Encapsulation Block

After you clone an Encapsulation block from the palette, you must create stubs for the block. You can create any number of input and output stubs of any type. GDA automatically names the stubs.

There are two ways of adding stubs to an Encapsulation block:

- Drag any path into the Encapsulation block from a block on the workspace.
- Use the Stub Tools palette to add any path to the block. For information on creating stubs using this technique, see "Stub Tools" on page 615.

The following figure shows an Encapsulation block with one data input port and two inference output ports:

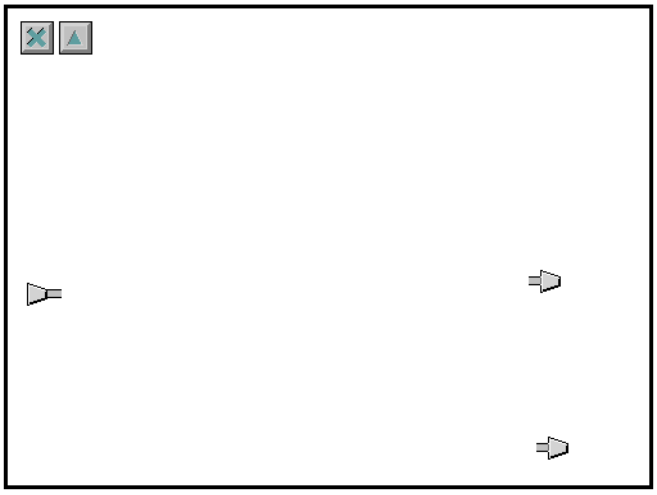


Specifying the Diagram for the Encapsulation

After creating stubs for the block, select view diagram from the encapsulation block's menu to display the **diagram** for the block. The diagram is a subworkspace of the Encapsulation block on which you specify the details of the encapsulation.

By default, the block contains connection posts for connecting to the superior workspace, a close button, and a go to superior workspace button, as this figure

shows. The number and type of connection posts corresponds to the number of input and output ports created for the block.



The diagram can include any number of blocks in any configuration. The inputs to the Encapsulation block passes to the block connected to the input connection post(s) on the diagram. This value then passes through all the blocks in the diagram until it reaches the output connection post(s). The output(s) of the Encapsulation block then passes to the next block in the diagram.

Caution Do not create connection posts manually on the diagram; always create connection stubs on the block first to create the connection posts on the diagram. Also, all connection posts in the diagram must connect to the superior block.

Converting an Encapsulation Block to a Single Source Encapsulation

You can convert an Encapsulation block to a Single Source Encapsulation block using the convert to sse menu choice. Selecting this menu choice prompts you to enter the name of the new subclass, and then creates a new subclass of Single Source Encapsulation block. You can edit the subclass by using the wizard, described in "Editing an Existing Custom Subclass" on page 192 in the *GDA User's Guide*.

Note You can only convert an Encapsulation block to an SSE if the subworkspace contains blocks alone. If you attempt to convert an Encapsulation block that contains G2 items, such as rules, graphs, or charts, GDA displays a message indicating that this is not possible. To convert the block, first delete the G2 items from the subworkspace.

Configuring

The Encapsulation block has no configurable attributes.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Single Source Encapsulation	page 208	*User's Guide*

21

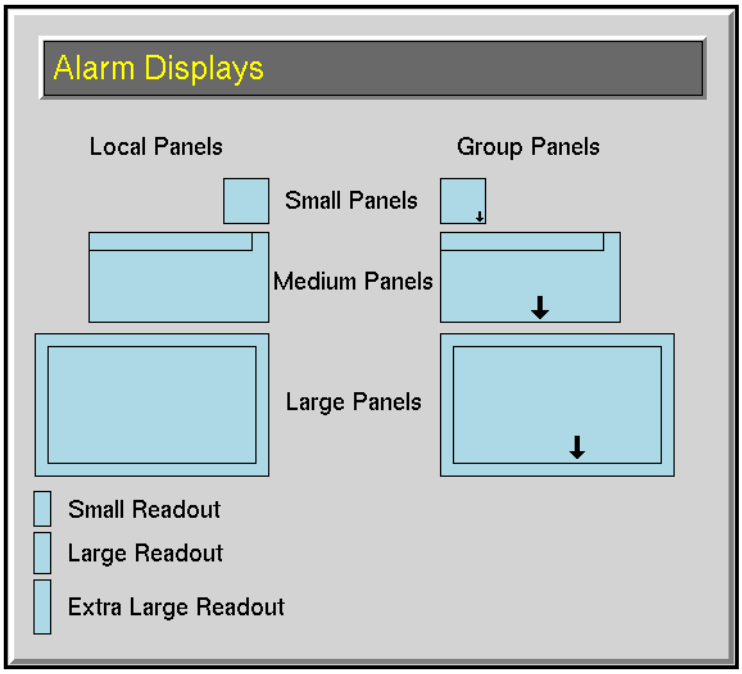
Alarm Displays

Describes the displays that you use to view the status of an alarm.

Introduction

Alarm displays let you view the status of an Alarm Capability, even when the capability is not visible. They are especially useful when your diagram is spread over several workspaces and you want to view and control all its alarms from one central workspace. They are also useful in smaller applications, since they are larger and more noticeable than an Alarm Capability.

You can find the Alarm Displays palette under the Other submenu of the Palettes menu:



Alarm Panels

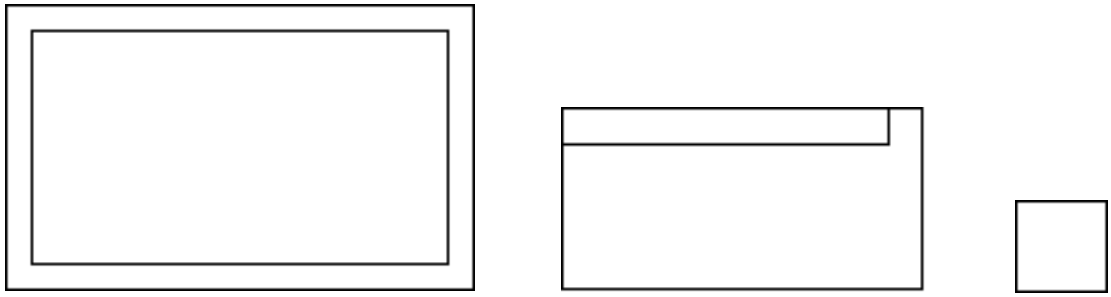
These displays let you monitor and manipulate an Alarm Capability:

- The Local Panels on page 485 let you work with one Alarm Capability.
- The Group Panels on page 495 let you work with several Alarm Capabilities at one time.

Readouts

The Alarm Readouts on page 498 let you monitor both an alarm's status and a block's value.They display two pieces of information: a block's output value and the alarm color of any active downstream alarm. They are useful when you want to show that a block's value is causing an alarm.

Local Panels



A Local Alarm Panel lets you monitor and manipulate an alarm. They come in three sizes: large, medium, and small. Color regions in the alarm panel reflect the severity of past and present alarm messages. Indicators in the alarm reflect whether there is a recurring alarm, whether the condition attached to the alarm is overridden, and more. The alarm panel's menu contains choices that let you acknowledge the alarm, view the alarm messages, override the condition, and more.

Local Alarm Panels are especially useful when your diagram is spread over several workspaces and you want to view and control all its alarms from one central workspace. They are also useful in smaller diagrams when you want to draw attention to an alarm with something larger than the Alarm Capability icon.

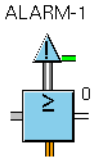
You can use Group Alarm Panels to group several alarm panels together so you can monitor and manipulate several alarms from one panel. For more information, see "Group Panels" on page 495.

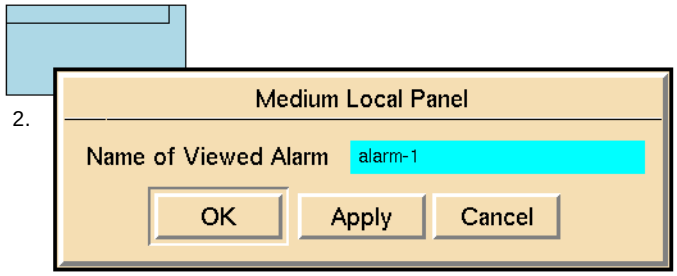
Identifying the Alarm

Before you can use a Local Alarm Panel, you must name the alarm and associate the named alarm with the panel.

To identify the alarm:

1. Create an Alarm Capability on a block, and name the alarm to which the alarm panel refers by choosing the name menu choice.
2. Create an Alarm Panel and configure the attribute Name of Viewed Alarm to refer to the name of the Alarm Capability.

1. 

2. 

Note If you enter an incorrect name into the Name of Viewed Alarm attribute for a Local Alarm Panel, GDA displays an error message in the error queue and an "E" on the panel. To remove the "E" from the panel, reset the panel.

Using Alarm Memory

A Local Alarm Panel can remember the severity of the last unacknowledged message for its associated alarm. To enable an alarm's memory, set the attribute Memory Enabled in the Alarm Capability to yes.

An unacknowledged message happens when an alarm becomes active and inactive without ever being acknowledged. They are possible if you leave the station when an alarm occurs. If you enable an alarm's memory, the associated alarm panel lets you know that the alarm happened while you were out so you can handle it. To see how an unacknowledged message affects an alarm panel, see "Reading the Panel" on page 486.

When an alarm's memory is enabled, you may also be able to get an explanation for the unacknowledged message. For more information, see "Remembering an Old Explanation" on page 492.

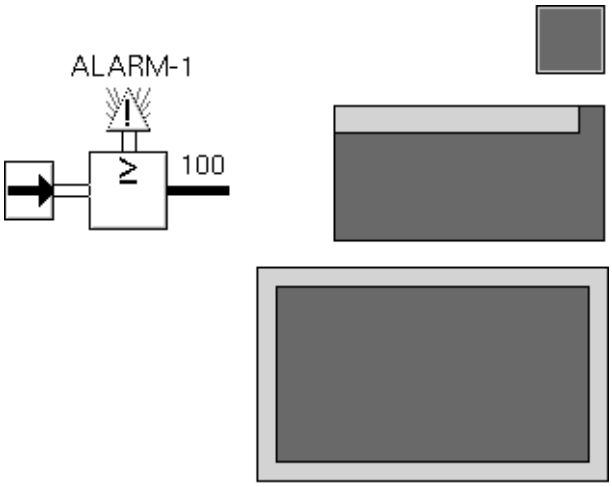
Reading the Panel

A Local Alarm Panel changes its color to reflect the severity of its associated Alarm Capability. It has two regions: status and border. The status region tells you the severity of the current alarm, if there is one. The border region tells you whether there is an unacknowledged alarm. This section explains what alarm panels for active and inactive alarms look like.

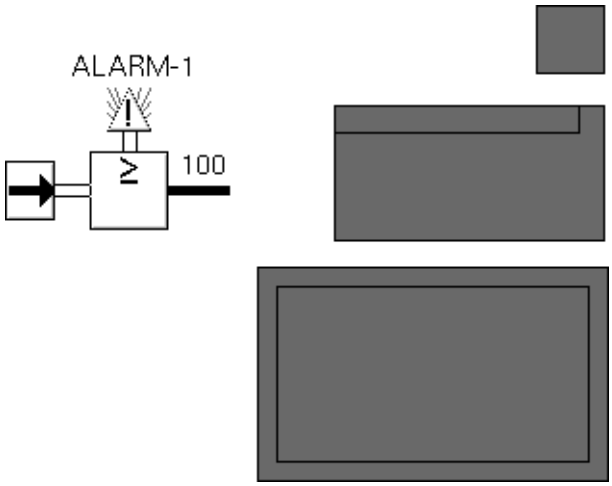
A Local Alarm Panel also has indicators on it to reflect its own status, the status of its associated alarm, the status of the condition attached to that alarm. This chapter explains how recurring alarms and alarm panel errors affect the alarm panel. For more information on indicators, see "Inhibiting the Panel" on page 490, "Locking" on page 491, and "Overriding" on page 492.

Viewing Active Alarms

When the alarm is active, the status region takes on the alarm's severity color. The border region is yellow until you acknowledge the alarm. Then it becomes the same color as the status region. For example, the following figure shows alarm panels for an active alarm that you have not acknowledged.



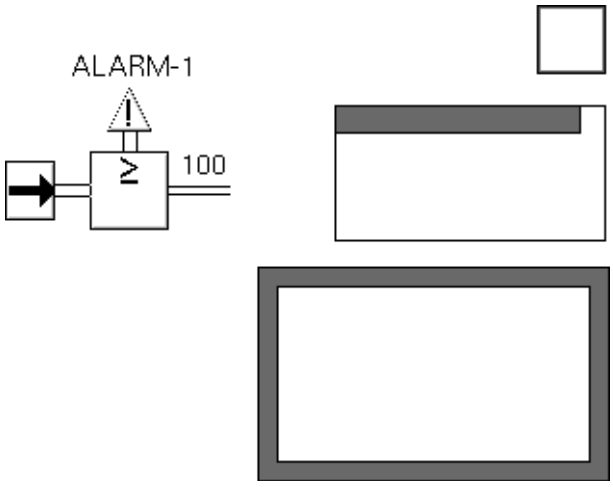
The following figure shows alarm panels for an active alarm that you have acknowledged.



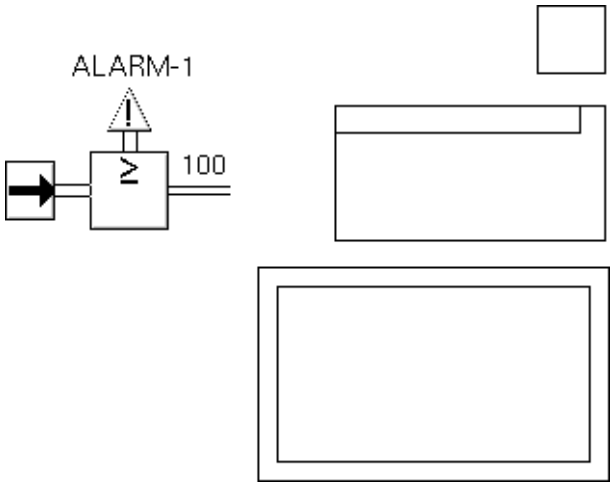
Viewing Inactive Alarms

When the alarm is inactive, the status region takes on a neutral color, usually light blue. The color of the border region depends on whether you are using alarm memory. If you are using alarm memory, the border takes on the severity color for the last unacknowledged alarm. If you are not using alarm memory, or if there is no unacknowledged alarm, the border region takes on the same color as the

status region. For example, the following shows alarm panels for an inactive alarm that has an unacknowledged message.

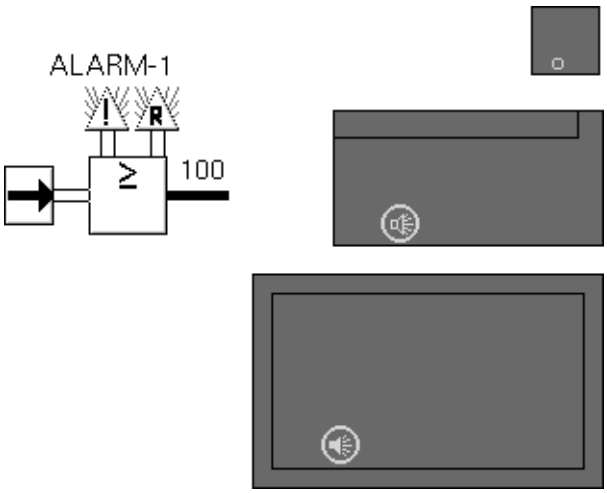


The following figure shows alarm panels for an inactive alarm that has no unacknowledged messages:



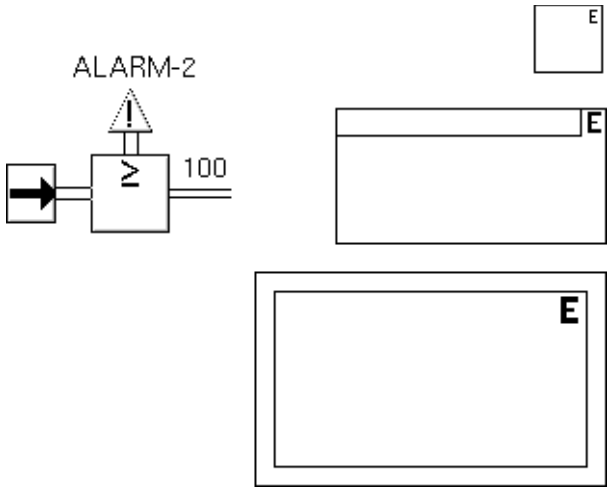
Viewing Recurring Alarms

If a Recurring Alarm Capability is active and it is attached to the same condition as an alarm panel's associated alarm, the alarm panel displays a recurring alarm indicator, as shown in this figure:



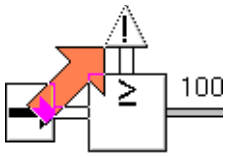
Viewing an Alarm Panel with an Error

If an alarm panel is not set up properly, it displays an error indicator, as shown in this figure:



Viewing the Alarm for an Alarm Panel

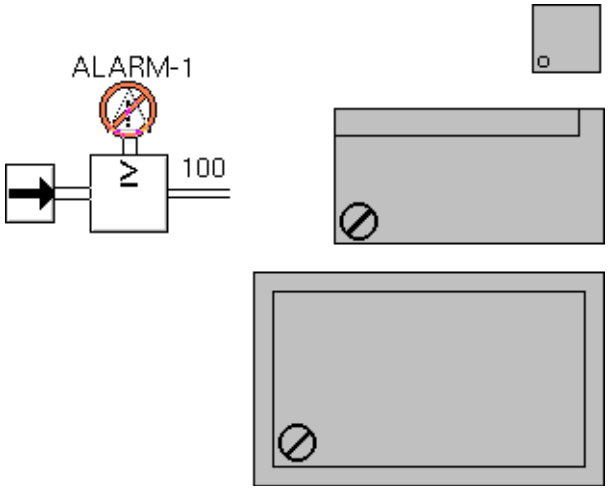
To view the alarm that a Local Alarm Panel refers to, choose go to alarm from the panel's menu. GDA shows the workspace of the alarm and places an arrow near the alarm for 15 seconds, as shown in the next figure. This choice is especially useful if an alarm and its alarm panel are on different workspaces.



Inhibiting the Panel

If the sensor for an alarm is broken or if the equipment associated with the alarm is off-line, you may want to inhibit the alarm to stop spurious alarms. To stop an alarm from updating, choose inhibit from the alarm panel's menu or its alarm's menus.

When an alarm panel is inhibited, it becomes completely gray, including the border, and a crossed out circle appears in the panel, as shown in the next figure. The alarm panel's body and border do not change color until you uninhibit the panel. The alarm associated with the alarm panel is covered with a crossed-out circle. The alarm still displays stress marks when it is active and creates and updates messages, but it does not display the alarm message queue, even if Show Messages is yes.



To let an alarm panel update its status again, choose uninhibit from the alarm panel's menu or one of its alarms' menus.

Acting on the Alarm

A Local Alarm Panel's menu contains two menu choices that act on the alarm that is associated with the panel. These choices let you acknowledge an alarm and view its messages.

Acknowledging Alarms

To acknowledge an alarm message, choose acknowledge alarm from the menu for the alarm panel associated with the alarm. Choosing this menu choice is the same as choosing acknowledge alarm from the alarm's menu.

To acknowledge a recurring alarm message, choose acknowledge recurring alarm from the menu for the alarm panel associated with the alarm. Choosing this menu choice is the same as choosing acknowledge recurring alarm from the alarm's menu.

Showing Alarm Messages

To view the messages for an alarm, choose show alarm messages from the menu for the alarm panel associated with the alarm. This menu choice is the same as choosing show alarm messages from the alarm's menu.

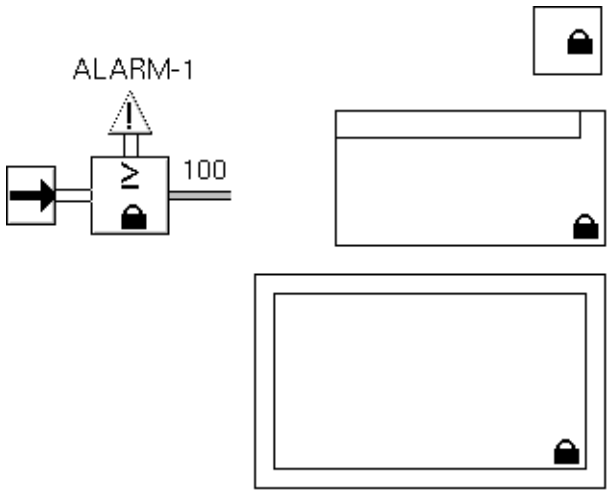
Acting on the Condition

A Local Alarm Panel's menu contains two menu choices that act on the condition that is attached to the associated alarm. These choices let you lock the condition, override it, and view explanations for its past and present status.

Locking

To lock a condition, choose lock from the menu of the alarm panel associated with the alarm attached to the condition. This menu choice is the same as choosing lock from the condition's menu.

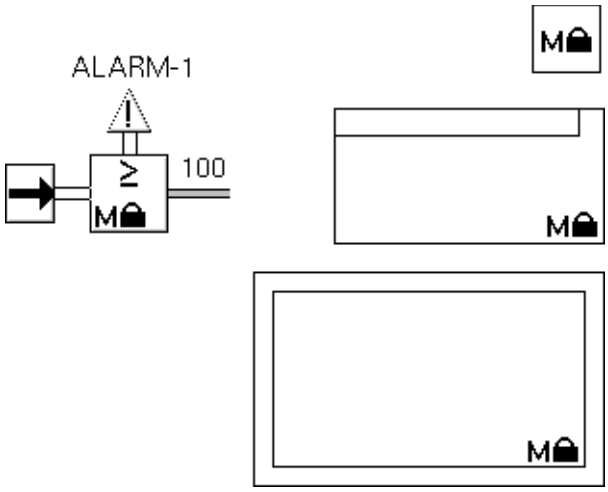
The following figure shows what Local Alarm Panels look like when you lock the conclusion associated with them:



Overriding

To override a condition, choose override from the menu of the alarm panel associated with the alarm attached to the condition. GDA displays an override dialog to let you enter a value. This menu choice is the same as choosing override from the condition's menu.

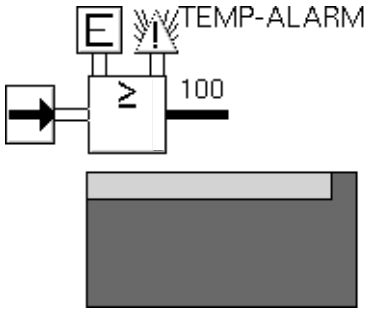
This figure shows what Local Alarm Panels look like when you override the conclusion associated with them:



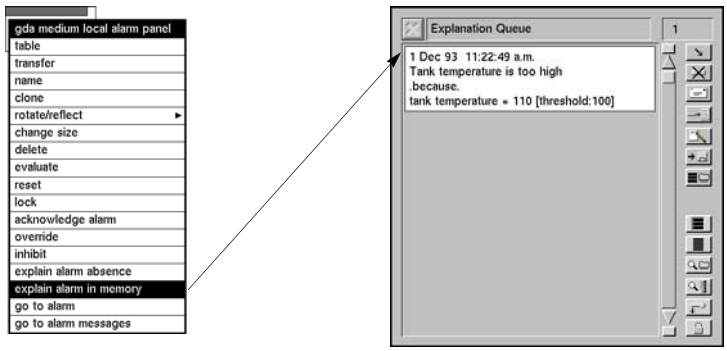
Remembering an Old Explanation

To remember why an alarm was raised even after it is lowered, attach both Alarm and Explanation Memory capabilities to the same block and set their Trigger On attributes to the same value. GDA notices that you want to remember the reason for the alarm, and adds the menu choice explain alarm in memory to the Alarm Panel for the Alarm. This menu choice is the same as choosing explanation of last true, explanation of last false, or explanation of last unknown, depending on the value of Trigger On.

For example, the Explanation Memory Capability in this figure remembers the reason for the last alarm. The Trigger On attributes in both the Explanation Memory Capability and the Alarm Capability are set to .true. The Alarm Panel below the High Value block is for the alarm.



To see the saved explanation, select explanation of last true from the High Value block, or you can choose explain alarm in memory from the Alarm Panel's menu, as shown in this figure:

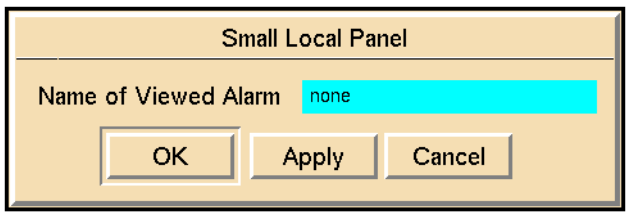


Explaining Current Status

You can view an explanation of an alarm panel's current state by choosing explain alarm or explain alarm absence from the Local Alarm Panel's menu. If the alarm for the Local Alarm Panel is active, explain alarm is in the panel's menu. If the alarm for the Local Alarm Panel is inactive, explain alarm absence is in the panel's menu. Choosing these menu choices is like choosing current explanation from the condition's menu.

Configuring

This is the configuration panel for a Small Local Alarm Panel. The configuration panel for a Medium or Large Local Alarm Panel is the same except for the block name.



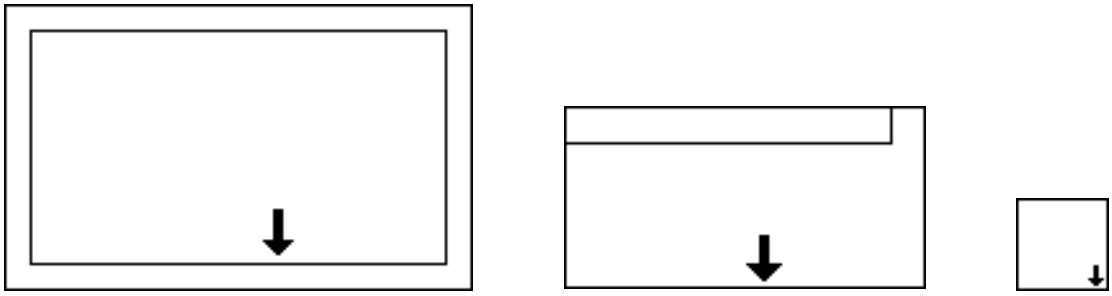
Attribute	Description
Name of Viewed Alarm	The name of the Alarm Capability whose alarm status the Alarm Panel displays.

See Also

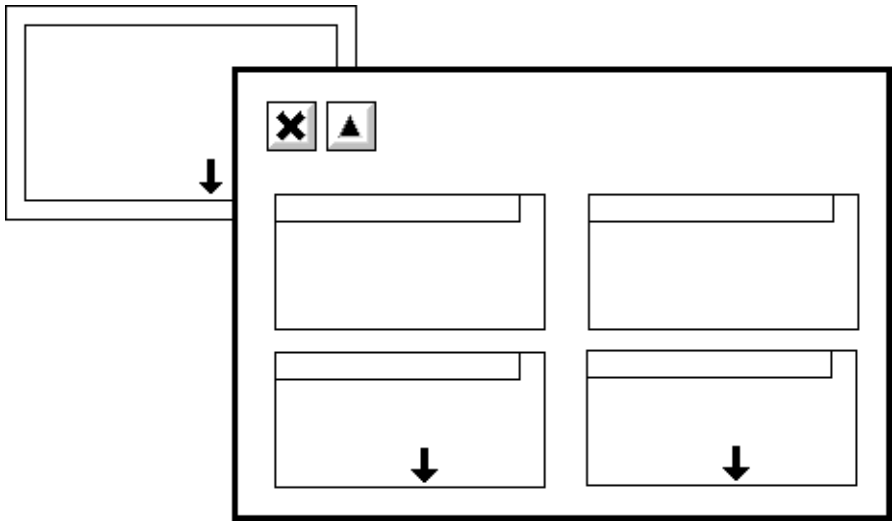
For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Group Alarm Panels	page 495	*Reference Manual*
The Alarm Readouts	page 498	*Reference Manual*
The Alarm, Recurring Alarm Capabilities	page 519	*Reference Manual*

Group Panels



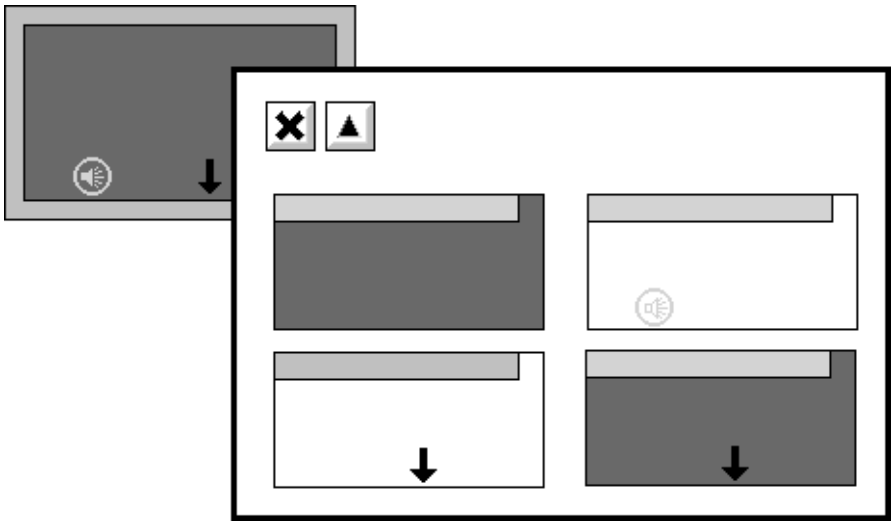
A Group Alarm Panel lets you monitor and manipulate several alarms from one alarm panel. You place Local Alarm Panels and other Group Alarm Panels onto its subworkspace, as shown in this figure. It displays the changes in those alarm panels.



Group Alarm Panels are especially useful when you have a lot of related alarms and you do not want to be overwhelmed with details. Since you can place a Group Alarm Panel on another Group Alarm Panel's subworkspace, you can organize the alarm panels hierarchically. By hiding details and organizing alarm panels intelligently, you can reduce clutter and draw attention to alarms more effectively.

The Group Alarm Panels look identical to Local Alarm Panels, except the group panels have an arrow on them, indicating that they have a subworkspace. The status region of the Group Alarm Panel is the color for the highest severity of the status regions shown on its subworkspace. Similarly, the border region is the color for the highest severity of the border regions shown on its subworkspace. If any of the panels on the subworkspace have an indicator visible (such as the lock or recurring alarm indicators), the indicator is visible on the Group Alarm Panel.

This figure shows a Group Alarm Panel that has several active alarm panels on its subworkspace:



Viewing Details

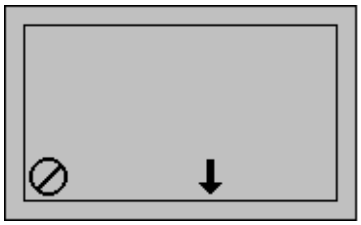
To see the alarm panels on a Group Alarm Panel's subworkspace, choose go to details from the Group Alarm Panel's menu. GDA displays the subworkspace.

Acknowledging Alarms

You can acknowledge all the alarms and recurring alarms displayed on a Group Alarm Panel's subworkspace. To acknowledge all the alarms, choose acknowledge alarm from the Group Alarm Panel's menu. To acknowledge all the recurring alarms, choose acknowledge recurring alarm from the Group Alarm Panel's menu. Choosing these menu choices is like choosing acknowledge alarm or acknowledge recurring alarm from each alarm panel on the Group Alarm Panel's subworkspace.

Inhibiting Panels

You can prevent a Group Alarm Panel and all the alarm panels on its subworkspace from updating their status. This is especially useful when there are a lot of alarm panels, and you want to prevent some panels from updating their status so you can concentrate on the others. To stop a Group Alarm Panel from updating, choose inhibit from its menu. Choosing this menu choice is like choosing inhibit from each alarm panel on the Group Alarm Panel's subworkspace.



Uninhibiting Local Alarms

The Group Panels sometimes contain a menu choice for uninhibiting local alarms contained on the Group Panel subworkspace. If you select inhibit for one or more local alarms on the Group Panel, the uninhibit grouped alarms choice appears in the menu for the Group Panel. Selecting this menu choice is equivalent to selecting uninhibit for each inhibited local alarm on the Group Panel.

Configuring

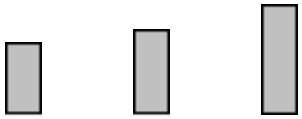
Group Alarm Panels have no configurable attributes.

See Also

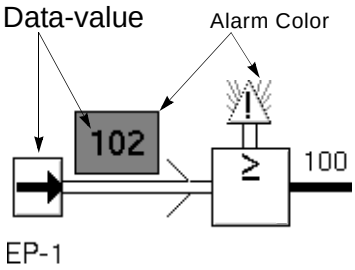
For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
The Local Alarm Panels	page 485	*Reference Manual*
The Alarm Readouts	page 498	*Reference Manual*
The Alarm and Recurring Alarm Capabilities	page 519	*Reference Manual*

Alarm Readouts



An Alarm Readout displays two pieces of information: a block's output value and the alarm color of any active downstream alarm. It is useful when you want to show that a block's value is causing an alarm or when you need to display the value of a block that does not have a dp-out or ip-out attribute. To display a block's value, set the attributes Name of Viewed Block, Portname, and Display Attribute. To display the alarm color of a downstream alarm, set the alarm's Update Alarm Readouts attribute to yes.



When you clone an Alarm Readout, GDA also performs the following operations:

- After cloning, GDA immediately links the Alarm Readout with its data source. You do not have to configure the readout to create this link.
- The color of the cloned Alarm Readout is the permanent color of the source Alarm Readout object, not the color of the object from which you are cloning.

Identifying the Readout

Before you can use a Readout, you must name the block whose value the readout will display, and associate the named block with the readout.

To identify the readout:

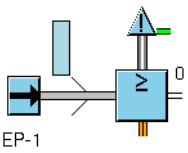
1. Name a block whose output value you want to display by choosing the name menu choice.
2. Create a Readout and place it next to the block.
3. Configure the attribute Name of Viewed Block to refer to the name of the block.
4. Configure the attribute Portname to name the correct output port of the block.

5 Configure the attribute Display Attribute to name the path attribute whose value you want to display.

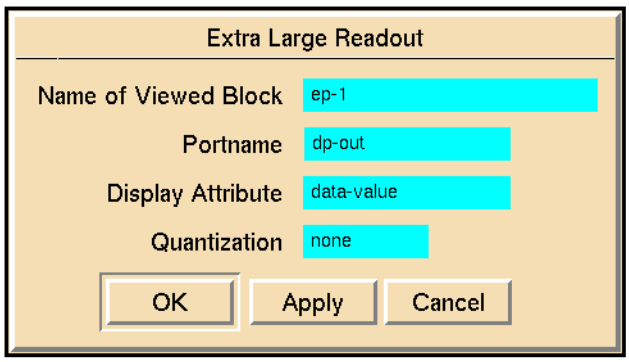
For example, if you are displaying a data path's value, Display Attribute is data-value, and if you are displaying an inference path's value, it should be belief-value.

6 If you want the readout to reflect the alarm color for a downstream alarm, make sure the alarm's Update Alarm Readouts attribute is set to yes.

1., 2.



3., 4., 5.



Rounding the Displayed Value

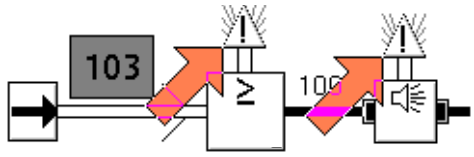
To round the value that a readout displays, set the Quantization attribute. The readout rounds the number to the nearest multiple of the quantization. For example, with a Quantization of 0.1, the readout converts 0.248 to 0.2 and 0.817 to 0.8. with a Quantization of 0.25, the readout converts 0.248 to 0.25 and 0.817 to 0. 75.

Going to Data Source

To view the block whose output path an Alarm Readout is displaying, choose go to data source from the Alarm Readout's menu. GDA brings the block's workspace to the front and puts an arrow next to the block for 15 seconds. This menu choice is especially useful if the block is on a different workspace from the readout.

Going to Alarms

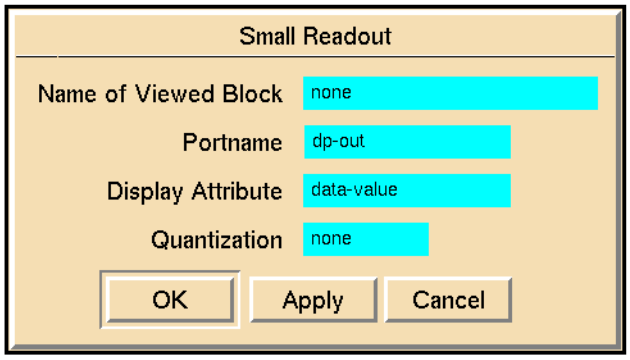
To view the active alarms downstream from an Alarm Readout, choose go to alarms. GDA brings the alarm's workspace to the front and puts an arrow next to the alarm for 15 seconds. This menu choice is especially useful if the alarm is on a different workspace from the readout. Note that this menu choice does not display inactive alarms upstream from the readout.



EP-1

Configuring

This is the configuration panel for a Small Readout. The panels for the other readouts are identical except for their names.



Attribute	Description
Name of Viewed Block	The name of the block whose output path value the Readout displays.
Portname	The name of the output port whose value the Readout displays.
Display Attribute	The name of the path attribute whose value the Readout displays.
Quantization	A number that determines how the readout rounds it displayed value.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Local Panels	page 485	*Reference Manual*
Group Panels	page 495	*Reference Manual*
The Alarm, Recurring Alarm Capabilities	page 519	*Reference Manual*
Belief Displays	page 505	*Reference Manual*

22

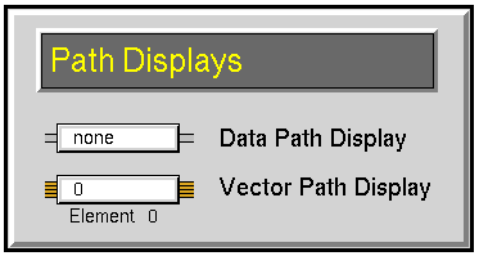
Path Displays

Describes the blocks that display data path values and inference path belief values.

Introduction

The blocks in the Path Displays palette let you display an inference path's belief value.

You can find the Path Displays palette in the Other submenu of the Palettes menu:

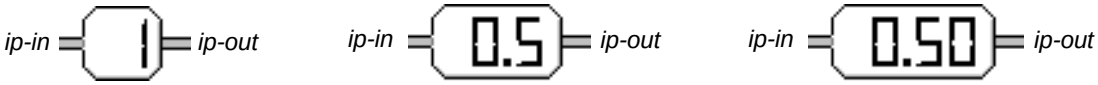


See Also

These blocks perform similar functions:

- The Alarm Readouts on page 498 display the value for a block's attribute. They are on the Alarm Displays palette in the Other menu.
- The Counters & Timers palette contains several blocks that also have readouts. The blocks can delay, time and count inference values. The Counters & Timers palette is in the Inference Blocks menu.

Belief Displays



A Belief Display block displays the belief value of its input inference path on its icon. This table describes how to choose the proper Belief Display:

If you want to see…	Use this display…
Whether a status value is .true or .false	'D' Belief Display
Whether a status value is .true, .false, or unknown	'D.D' Belief Display
A belief value with 1-digit accuracy	'D.D' Belief Display
A belief value with 2-digit accuracy	'D.DD' Belief Display

If a Belief Display cannot display all the digits in its input value, it rounds the value up. For example, a 'D.D' Belief Display would display 0.47 as 0.5, 0.23 as 0.2, and 0.75 as 0.8.

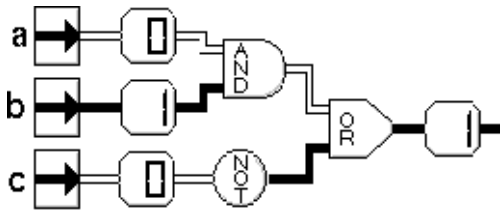
Note Do not use the 'D' Belief Display if you need to see whether the status value of a path is unknown. It displays the belief values 0.5 and 1.0 as 1.

Configuring

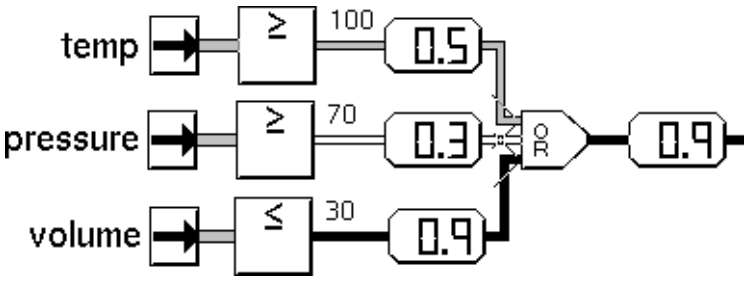
Belief Display blocks have no configurable attributes.

Example

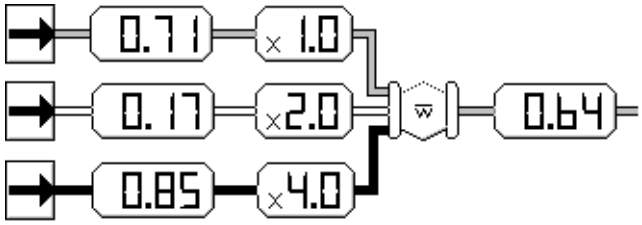
In this figure, 'D' Belief Displays show you the status values of the entry points a, b, and c and the value of the expression (a AND b) OR (NOT c):



In this figure, 'D.D' Belief Displays show you the belief values of three observations that use the Threshold Uncertainty attribute:



In this figure, 'D.DD' Belief Displays show you how a Weighted Evidence Combiner computes its value:



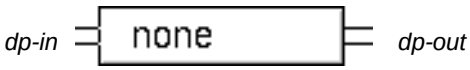
See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Displaying the Value on a Path	page 29	*User's Guide*
The Inference Delay gate	page 363	*Reference Manual*
The Persistence Gate	page 366	*Reference Manual*
The Timer gate	page 369	*Reference Manual*
The Inference Counter block	page 372	*Reference Manual*
The Control Delay block	page 441	*Reference Manual*

For more information on...	See...	In this book...
The Control Counter block	page 449	*Reference Manual*
The Alarm Readouts	page 498	*Reference Manual*

Data Path Display

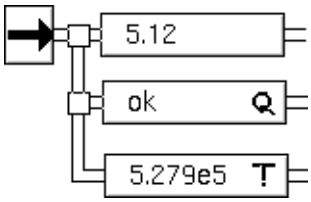


The Data Path Display block displays the value, quality, or collection time of its input value. By default, it displays the value. When it is displaying quality, a *Q* appears on the icon. When it is displaying collection time, a *T* appears on the icon.

Determining Which Path Attribute to Display

Use the menu choices show collection time, show value, and show quality, to display the path's Collection-time, Data-value, and Quality attributes, respectively.

The diagram below uses three Data Path Displays to display the value, quality, and collection time of one Entry Point.

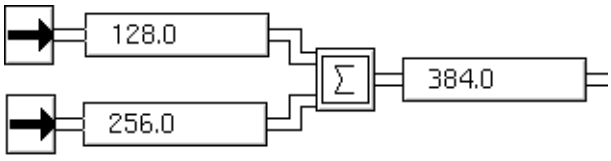


Configuring

The Data Path Display block has no configurable attributes.

Example

In the diagram below, Data Path Displays show an addition in progress.

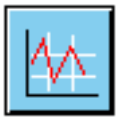


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Displaying the Value on a Path	page 29	*User's Guide*

Trend Chart

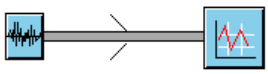


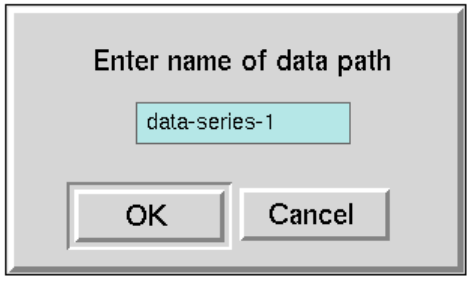
The Trend Chart block enables GDA to use trend charts, which are a standard feature of G2. Trend charts provide a graphical display that plots time series or historical data over a designated period of time, plotting the collection time of the data against the data value. A trend chart plots the history of a variable or parameter.

The Trend Chart block is a stubless peer input block enabling you to attach any number of data paths to the Trend Chart block. You can use the Trend Chart block to plot the values of all the attached data paths on a trend chart.

To create a trend chart block on a workspace:

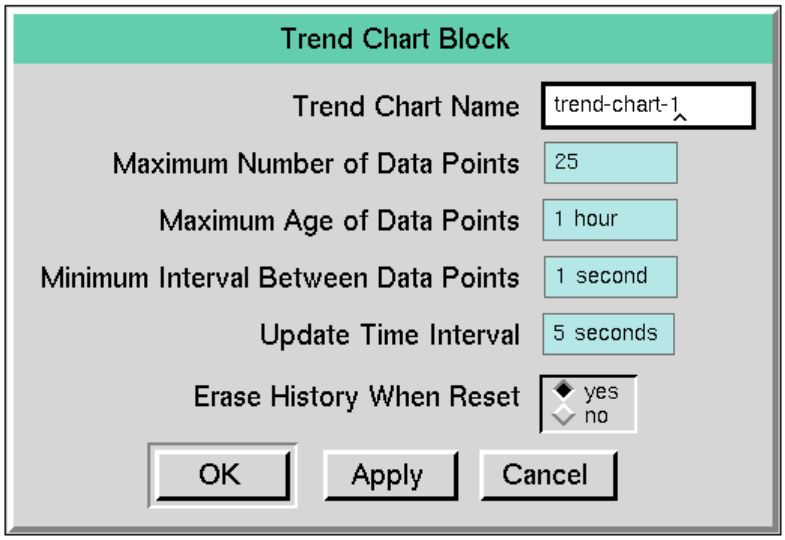
1. Create a trend chart by using the KB Workspace > New Display > trend-chart G2 menu choice.
2. Name the trend chart, for example, TREND-CHART-1.
3. Clone a Trend Chart block by using the Palettes > Other > Path Displays palette in the GDA menu bar.
4. Connect an input, for example, a white noise generator, via a data path to the Trend Chart block, which pops up the Data Path Name dialog, as shown in the following figure:





5. Enter the data path name and click OK. The corresponding plot created in the trend chart has the same name as the data path.

   Each data path connected to the Trend Chart block represents a plot. The Trend Chart block can only process data paths that are outputs of GDAblocks.
6. Configure the Trend Chart block by using the configuration dialog, which appears after you name the data path, as shown in the following figure:

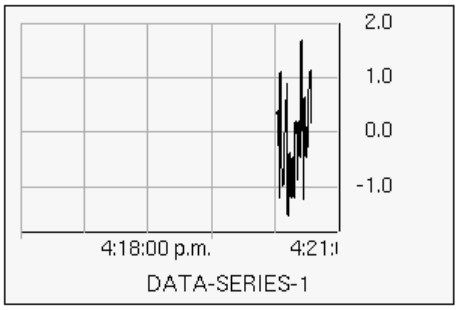


7 Enter the trend chart name, for example, TREND-CHART-1. The minimum configuration requires that a trend chart have a name.

Specify the history for the data path by using the values of the following attributes:

Attribute	Description
Trend chart name	The name of the trend chart upon which the values are to be plotted.
Maximum number of data points	Maximum number of data points to be stored in history.
Maximum age of data points	Maximum age of data points to be stored in history.
Minimum interval between data points	Minimum interval between two data points in history.
Update time interval	The time interval after which each plot evaluates the expression in its expression attribute. The update-time-interval attribute sets the display-update-interval attribute on the time-axis.
Erase history when reset	History is either retained upon reset or erased. By default, history is erased upon reset.

The resulting trend chart is shown in the following figure:



A plot, like the one in the preceding figure, is a component of a trend chart. A trend chart can have one or more plots. The attributes in the plot subtable that are controlled by the Trend Chart block are described in the following table:

Attribute	Description
Name	The name of a plot is set as the name of the corresponding port.
Use-local-history?	Set to no
Update-interval	The update-interval gets its value from the update time interval attribute in the configuration dialog.
Expression	Set by the Trend Chart block.

Except for the trend chart configuration dialog update-time-interval attribute, which sets the display-update-interval attribute on the time-axis, the Trend Chart block controls only the plots of a trend chart. You can change all other settings such as value-axis, time-axis, point-formats, connector-formats, and trend-chart-format-subtable in the trend chart attribute tables.

Note In the plot subtable of a trend chart, do not add a plot, delete a plot, or change the values of the names, use-local-history?, update-interval, and expression attributes. The Trend Chart block configuration takes care of these attributes.

To locate the trend chart associated with a Trend Chart block:

➔ Choose the Go-to-trend-chart user menu choice on a Trend Chart block, which draws an arrow next to the trend chart and displays its subworkspace, if any.

To locate the data path for a specified plot on a trend chart:

➔ Choose the Show-data-path-of-plot user menu choice on a trend chart, which draws an arrow next to the data path and displays its subworkspace, if any.

To change the name of a trend chart plot and its associated data path:

➔ Choose Edit-plot-name user menu choice on a trend chart, which displays the Data Path Name dialog.

If there is more than one data path in the trend chart, you can choose the appropriate data path in the Select Plot Name dialog.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Displaying the Value on a Path	page 29	*User's Guide*

23

Capabilities and Restrictions

Describes the special blocks that add features to blocks, such as alarming, graphing, or logging.

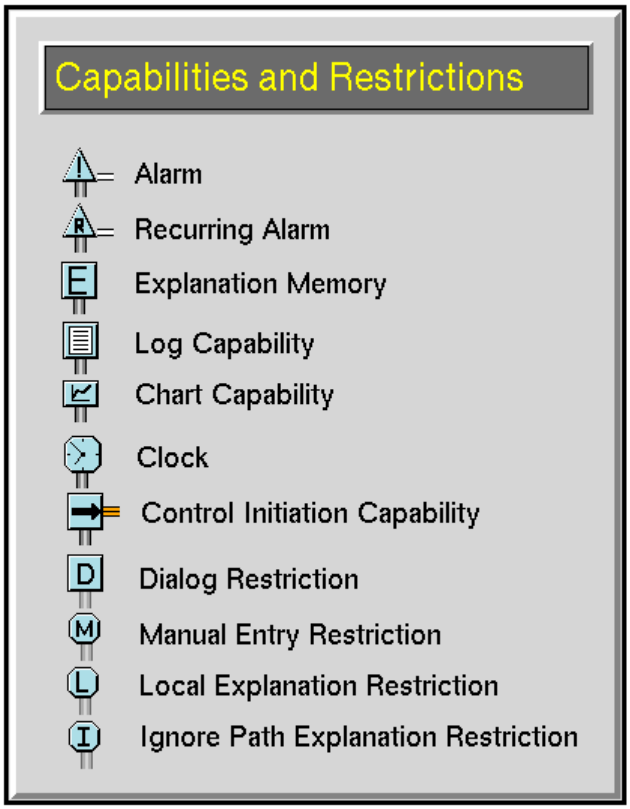
Introduction

Capabilities are special blocks that let you add features and attributes to other GDA Blocks. They attach directly to the top of another block. Think of a Capability as a "cap": a small feature on top of a block that adds something to it. For example, a Log Capability lets a block write a history of the values it receives to a file.

Restrictions are a type of Capability that restrict a block's actions. For example, a Dialog Restriction restricts the prompts used in the Override dialog for a block, and an Ignore Path Explanation Restriction prevents a block from being included in an explanation.

You can only attach capabilities and restrictions to a single block.

You can find the Capabilities and Restrictions palette under the Other submenu of the Palettes menu:



Raising Alarms

These capabilities draw attention to dangerous conditions. When its attached block passes a certain value, the capability changes appearance and sends a message to a message queue.

- The Alarm Capability on page 519 warns you about a condition that has occurred once.
- The Recurring Alarm Capability on page 519 warns you about conditions that have occurred repeatedly. It must be used together with an Alarm Capability.

Specifying Explanations

GDA can generate an explanation for a condition's status value by analyzing the arrangement of the inference blocks in a diagram. These capabilities let you control how GDA analyzes those blocks:

- The Explanation Memory block on page 533 remembers the explanation that the attached condition had when its input status was a certain value.
- The Local Explanation Restriction block on page 563 lets GDA know that the attached condition provides the explanation for this path.
- The Ignore Path Explanation Restriction block on page 565 removes a path from an explanation.

For more information on how GDA creates explanations, see "Specifying and Generating Explanations" on page 93 in the *GDA User's Guide*.

Charting Attributes

The Chart Capability on page 541 lets you display a block's data on a chart. It lets you configure the chart from a dialog and change how the data is displayed.

Getting Data from the Operator

These capabilities let you choose how GDA gets data from the operator:

- The Dialog Restriction block on page 555 lets you customize the prompts in a block's Override dialog.
- The Manual Entry Restriction block on page 559 lets GDA know that the attached block gets its input from the operator only, and not from within G2.

Logging Changes

The Log Capability block on page 537 lets you record the values that a block receives to a file.

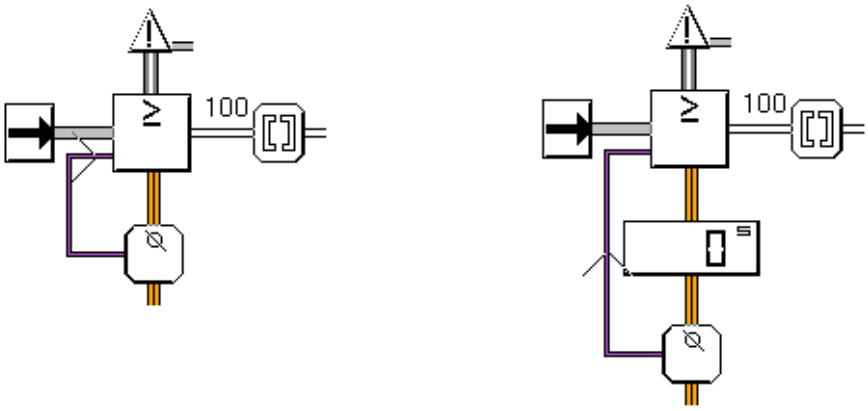
Forcing a Block to Evaluate

The Clock block on page 552 forces the attached block to evaluate at specified times.

Evaluating Capabilities and Blocks

GDA evaluates a capability after the block to which it is attached; however, this does not necessarily guarantee evaluation of the capability immediately after the block to which it is attached.

On rare occasions, this may produce unexpected results. In the example on the left, if GDA evaluates the Alarm Capability first, the alarm becomes active before the Reset block evaluates, which is the expected behavior. If GDA evaluates the Reset block before the Alarm Capability, however, the Alarm Capability will become active, even if the block to which the alarm is attached passes .true. To insure that the Reset block evaluates after the Alarm Capability, include a Control Delay block, as in the second example.



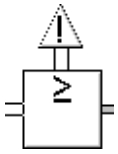
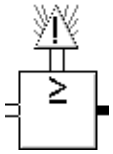
For more information, see “Understanding the GDA Block Evaluation Engine” on page 126.

Creating Capability and Restriction Links

Capability and Restriction Links are not the same; you attach a Capability to a block with a Capability Link, and you attach a Restriction to a block by using a Restriction Link. In other words, if you create a Capability and then delete it, leaving the Capability Link stub attached to the block, you cannot then drag a Restriction into the stub. You must delete the Capability Link first and then connect the Restriction.

Alarm, Recurring Alarm


An Alarm capability draws attention to a dangerous condition; for example, a tank's temperature is too high. When its attached block passes a certain value, the capability changes appearance and sends a message to a message queue. It changes appearance by drawing stress lines around its icon, as shown in this figure:

You can use a Recurring Alarm capability together with an Alarm capability to detect a recurring condition; for example, a tank's temperature has gone too high more than three times in the past hour. It is useful when a condition may have a different cause when it occurs repeatedly. For example, if a tank's thermostat goes too high once, the heater is too hot, but if it goes too high several times an hour, the thermostat may be broken.

You can attach an Alarm or Recurring Alarm to these blocks:

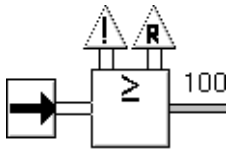
- Any block on the Observations palette.
- Any block on the Conditions palette.
- The Belief Entry Point block on the Entry Points palette.
- The Inference Output Action block on the Control Actions palette.
- The Network Belief Entry Point block on the Network palette.

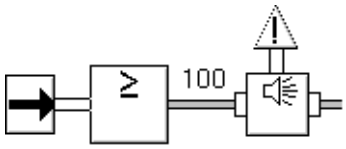
You can control when an Alarm capability or Recurring Alarm capability raises an alarm and what happens when it raises an alarm. For example, these capabilities can provide an explanation for the alarm, give advice on how to fix the condition that caused the alarm, and display its severity.

Do not use a Recurring Alarm capability alone. Use it only on a block that also has an Alarm capability. Think of it as adding its features and attributes to the Alarm capability instead of the block. If you want to raise an alarm only if a condition occurs repeatedly, use a Recurring Condition block with an Alarm capability.

For example, the diagram in the first figure below raises one type of alarm whenever the thermostat is too high and another type of alarm when the thermostat is too high more than three times an hour. The diagram in the second

figure raises an alarm only when the thermostat is too high more than three times an hour.





Optionally, you can add Alarm Panels, Alarm Readouts, and Alarm Groups to your application. They refer to an Alarm capability and reflect its status or let you manipulate it in different ways. An Alarm Panel changes color when its Alarm capability is active. An Alarm Readout can display the value of the block that is causing the alarm. An Alarm Group lets you enter data for entry points downstream from an Alarm capability.

An Alarm Panel, Alarm Readout, or Alarm Group cannot refer to a Recurring Alarm capability. Instead, let it refer to the Alarm capability that is attached to the same block as the Recurring Alarm capability.

Causing Downstream Actions to Occur When Acknowledging an Alarm

The Alarm and Recurring Alarm capabilities have an output inference path that turns .true when the user acknowledges the alarm and .false when the alarm is active and it is unacknowledged. Initially, the output inference path is unknown.

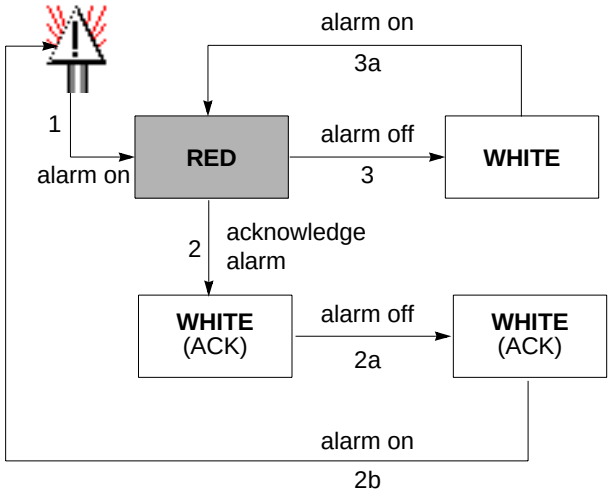
The output inference path causes downstream actions to occur when the user acknowledges the alarm. When used in combination with the Event Window Gate block on the Temporal Gates palette, for example, the output inference path can cause downstream actions to occur when the user fails to acknowledge the alarm within a given time window.

To add an inference path to an existing alarm capability, simply drag an inference path into the Alarm capability.

Understanding the Process of Receiving and Acknowledging Alarms

The following figure shows a flow diagram representing the process of receiving and acknowledging an alarm entry. The boxes represent alarm entries in the

queue and the arrows represent GDA or user actions. The labels refer to the steps following the figure. The colors red and white depend on the severity of the alarm and are specifiable by the user.



The steps in this process are:

1 The alarm becomes active, which causes GDA to send an alarm entry to the queue.

2 If the user acknowledges the alarm entry, the alarm entry turns white and includes (*ACK*) in the alarm entry text.

a If the alarm becomes inactive, the acknowledged alarm entry remains in the queue.

b If the alarm becomes active again, GDA sends a new alarm entry to the queue, and the process starts again. The acknowledged alarm entry remains in the queue.

3 If you do not acknowledge the alarm entry and the alarm becomes inactive again, the alarm entry turns white and remains in the queue.

   a If the alarm becomes active again, the color of the existing alarm entry changes from white to red and the process starts again. GDA updates the History attribute of the existing alarm entry using a new timestamp.

The following table summarizes this process:

If the user...	And the alarm becomes inactive and then active again, then...
Acknowledges an active alarm	GDA creates a new alarm entry in the queue, leaving the existing acknowledged alarm entry in the queue
Does not acknowledge an active alarm	GDA updates the History attribute of the existing alarm entry and does not create a new entry in the queue

Using Alarms and Recurring Alarms Together

You use an Alarm and a Recurring Alarm capability together on the same block.

If the Recurring Alarm capability goes on, GDA sends alarms entries to the queue according to the specification of the Recurring Alarm.

If the Alarm capability goes on at the same time as the Recurring Alarm, GDA does not send an entry to the alarm queue until the user resets the Recurring Alarm.

Specifying What Causes an Alarm

You can specify what conditions cause an alarm. With an Alarm capability, you just need to specify the output value for a block. With a Recurring Alarm capability, you also need to specify how often that value is passed.

With an Alarm Capability

To specify when an Alarm capability raises an alarm, use the attribute Trigger On. The capability creates an entry when the block to which it is attached passes the status value Trigger On. Trigger On can be .true, .false, or unknown.

With a Recurring Alarm Capability

To specify when a Recurring Alarm capability raises an alarm, use the attributes Trigger On, Recurring Alarm Threshold, and Recurring Alarm Interval. The capability creates an alarm entry when the attached block passes a certain status value more than a certain number of times in a specified period of time. Set the

attribute Trigger On to the status value to watch for. Trigger On may be .true, .false, or unknown. Set the attribute Recurring Alarm Threshold to the number of times the value must be detected. Set the attribute Recurring Alarm Interval to the period of time.

For example, if Trigger On is .true, Recurring Alarm Threshold is 10, and Recurring Alarm Interval is 1 hour, the capability creates an alarm if the attached block passes .true more than 10 times in one hour.

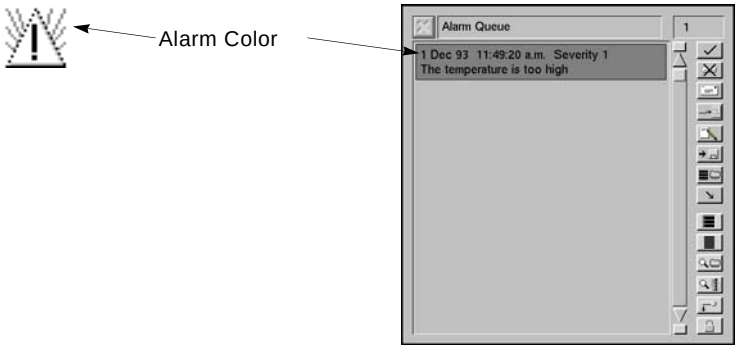
Specifying What Happens When an Alarm Occurs

You can specify what happens when an Alarm capability or Recurring Alarm capability raises an alarm. For example, the capability can set the alarm's severity, show the alarm's message queue, generate an explanation, and give advice.

Setting the Severity

To let the operator know how severe the alarm is, set the attribute Severity. Severity can be any number from 1 to 15, where 1 is most severe and 15 is least severe.

Each severity level has a color associated with it. GDA uses that color for the stress lines in an active alarm and as the background color for an unacknowledged alarm entry in a queue, as shown in this figure:



These are the default colors for the severity levels: 1 is red, 2 is orange, 3 is gold, and the rest are gray. You can change the colors with the Alarm Colors palette. To use the Alarm Colors palette, bring up the Configuration Tools menu bar, go to the Colors menu, and choose Alarm Colors.

Showing the Queue

An Alarm or Recurring Alarm capability can display the alarm messages whenever the operator needs to see a new one. If the attribute Show Messages is yes, the capability displays the alarm queue whenever it adds a new message to it. The capability does not display the queue when it updates a message.

The attribute Display Messages On specifies the queue that messages appear on and that the capability shows when Show Messages is yes.

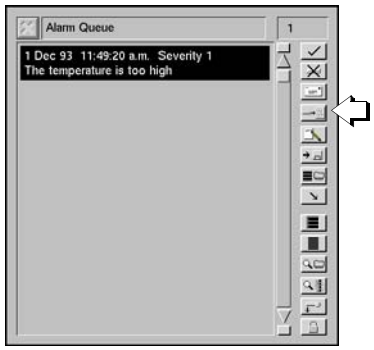
Generating an Explanation

By default, an Alarm Queue does not create an explanation for an alarm until you press the Explanation button in the View Entry dialog. If you want a capability to generate an explanation for an alarm immediately when the alarm is raised, set Generate Automatic Explanation to yes. This attribute is useful when the conditions that caused the alarm change quickly and you want the operator to see the explanation of the conditions right when the alarm was created.

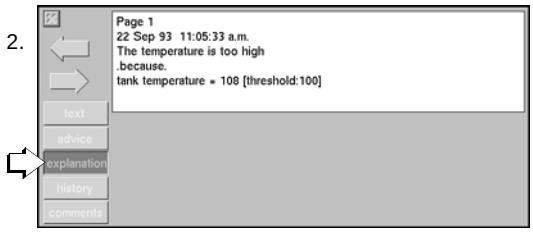
To view the explanation for an alarm message:

1. Select the message and press the View Entry button.
2. When GDA displays the View Entry dialog, press the Explanation button.

1.



2.



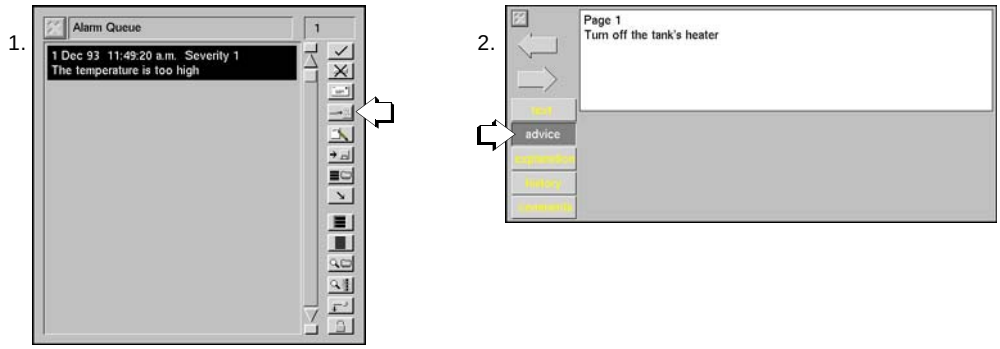
Giving Advice

You can give the operator advice on how to solve the problem that caused the alarm. Set the attribute Advice to a a string that contains your recommendation, such as "Turn off the heater" or "Check ventilation".

Note The attribute Advice can include the values of certain G2 expressions, as described in "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*. For example, the string "Shut down the [the contents of tank-1] tank" might appear as "Shut down the water tank", where water is the contents of tank-1.

To view the advice for an alarm message:

1. Select the message and press the View Entry button.
2. When GDA displays the View Entry dialog, press the Advice button.



3 When GDA displays the View Entry dialog, press the Advice button.

Choosing the Queue

You can choose which queue displays the messages for an Alarm or Recurring Alarm capability. For example, if you have a lot of alarms, you might want to display the messages from some alarms on different queues. Set the attribute Display Alarm Messages On to the name of the queue. By default, Alarm and Recurring Alarm capabilities display their messages on the Alarm Queue.

Setting the Type of Queue Entry

GDA use a G2 class to create messages in a queue. If you have created your own class for alarm messages and want an alarm to use it, set the attribute Entry Class to the name of your class.

Acknowledging Alarm Messages

Once an Alarm or Recurring Alarm capability raises an alarm, the operator needs to acknowledge it. To acknowledge an alarm from the capability, choose acknowledge alarm from the capability's menu.

Note You can also acknowledge an alarm by choosing acknowledge alarm from an Alarm Panel's menu or by pressing the Acknowledge button in the alarm message queue.

Deleting Unacknowledged Messages

Generally, you have to acknowledge an alarm before you can delete its message from the alarm message queue with the Delete Entries button. If you want to delete messages without acknowledging them first, set the attribute Require Acknowledgement to no.

Note that when Require Acknowledgement is yes, GDA can still delete unacknowledged messages in some cases. If you press the Flush Queue button in a message queue, the queue deletes all its messages, even unacknowledged ones.

And if the number of messages in a queue equals its Maximum Number of Entries attribute and it receives a new message, it deletes the oldest message, even if it is unacknowledged.

Viewing Alarm Messages

To display the queue that contains an alarm's messages, choose show alarm messages from the alarm's menu. GDA brings the message queue to the front.

Note You can also display an alarm's message queue by choosing go to alarm messages from a Local Alarm Panel's menu.

Using with Alarm Panels

If you use Alarm Panels with an Alarm capability, GDA adds an additional menu choice to alarms and lets you control whether the panel remembers an alarm's past status.

Note Alarm panels give you an easy way to monitor the status of alarms. There are two types: local and group. Local Alarm Panels let you monitor the current and past status of one alarm that may be on another workspace. Group Alarm Panels let you monitor the current and past status of several alarms in one small display. For more information, see "Local Panels" on page 485 and "Group Panels" on page 495.

Inhibiting Alarm Panels

If you have several Alarm Panels, you may want to prevent some panels from updating their status so you can concentrate on the others. To stop a panel from updating, choose inhibit from the alarm panel's menu or one of its alarms' menus. When an alarm's alarm panel is inhibited, the alarm updates and creates messages, but it does not display the alarm message queue, even if Show Messages is yes.

To let an alarm panel update its status again, choose uninhibit from the alarm panel's menu or one its alarms' menus.

Remembering the Last Unacknowledged Alarm

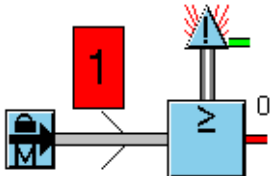
To have an Alarm Panel remember the most recent unacknowledged alarm, set the attribute Memory Enabled for the alarms associated with the panel to yes. The

Alarm Panel displays the severity color of the last unacknowledged alarm in the panel's border region and may let you view the explanation for the last unacknowledged alarm with the menu choice explain alarm in memory. For more information, see "Remembering an Old Explanation" on page 492.

Using with Alarm Readouts

When you use Alarm Readouts with an Alarm capability, you can choose whether the readouts' background colors match an alarm's severity color. To have an alarm update all the readouts that are downstream from it, set the alarm's Update Alarm Readout attribute to yes. To leave the readouts' background color alone, set Update Alarm Readout to no.

This example shows a readout whose color reflects the Severity of the alarm:



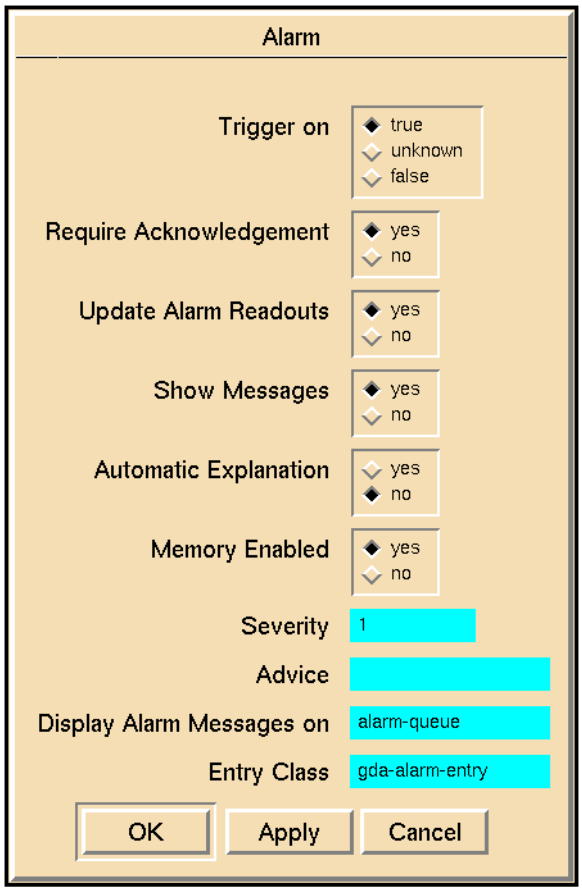
Note Alarm Readouts let you display the value of a block's attribute, in addition to an alarm's status. For more information, see "Alarm Readouts" on page 498.

Configuring

The Alarm capability and the Recurring Alarm capability have slightly different configuration panels.

Configuring the Alarm Capability

This is the configuration panel for the Alarm capability.

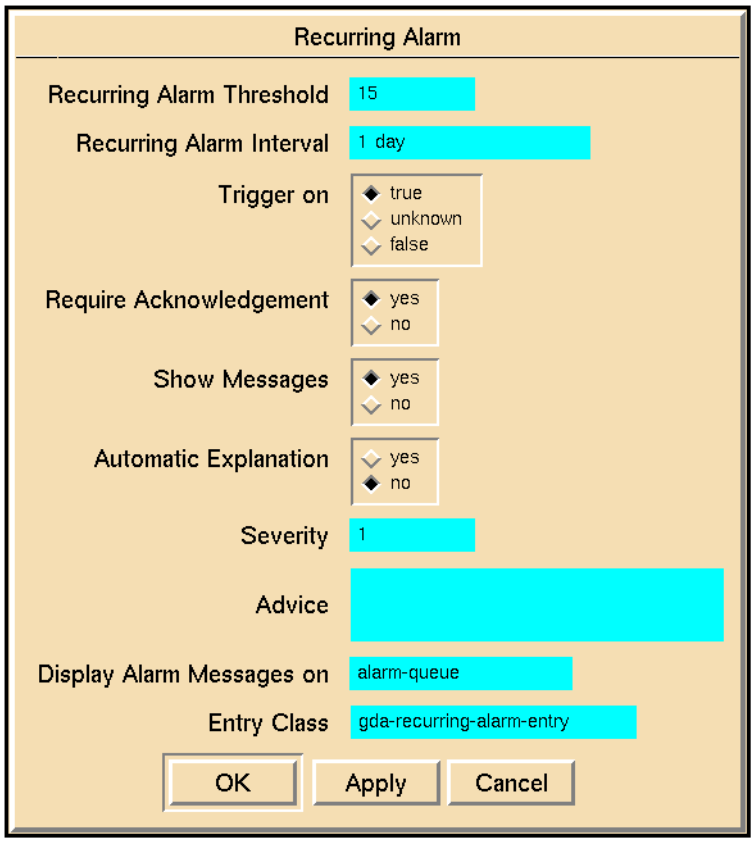


Attribute	Description
Trigger On	The truth value that causes the alarm to become active and the output path to turn . true.
Require Acknowledgement	Whether the alarm must be acknowledged before it is deleted.
Update Alarm Readouts	Whether the background color of a Readout associated with a block reflects the color associated with the Severity of the downstream Alarm capability.

Attribute	Description
Show Messages	Whether the Alarm capability displays the Alarm Queue whenever the alarm becomes active.
Automatic Explanation	Whether the Alarm capability automatically generates an explanation for the alarm at the same time that it generates an alarm message.
Memory Enabled	Whether an Alarm Panel remembers the last unacknowledged alarm that the Alarm capability generated.
Severity	A number between 1 (highest severity) and 15 (lowest severity) that indicates the severity of the alarm. The color of any Alarm Panel or Alarm Readout associated with the alarm, as well as the color of the message itself in the Alarm Queue indicates the severity of the alarm.
Advice	A string that provides advice on the alarm, which the operator can obtain via the Alarm Queue.
Display Alarm Messages On	The name of the queue on which to generate the alarm message.
Entry Class	The name of the message class that the queue uses to generate the alarm message.

Configuring the Recurring Alarm Capability

This is the configuration panel for the Recurring Alarm capability.

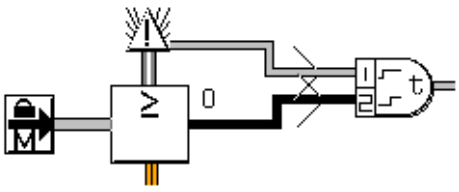


Attribute	Description
Recurring Alarm Threshold	The number of times that the alarm must occur before the capability generates an alarm message.
Recurring Alarm Interval	The time period during which the capability detects recurring alarms.
Trigger On	The truth value that causes the alarm to become active and the output path to turn . true.
Require Acknowledgement	Whether the alarm must be acknowledged before it is deleted.

Attribute	Description
Show Messages	Whether the Alarm capability displays the Alarm Queue whenever the alarm becomes active.
Automatic Explanation	Whether the Alarm capability automatically generates an explanation for the alarm at the same time that it generates an alarm message.
Severity	A number between 1 (highest severity) and 15 (lowest severity) that indicates the severity of the alarm. The color of any Alarm Panel or Alarm Readout associated with the alarm, as well as the color of the message itself in the Alarm Queue indicates the severity of the alarm.
Advice	A string that provides advice on the alarm, which the operator can obtain via the Alarm Queue.
Display Alarm Messages On	The name of the queue on which to generate the alarm message.
Entry Class	The name of the message class that the queue uses to generate the alarm message.

Example

The example below shows how to test whether an alarm has been acknowledged within a given time period after the alarm is triggered. The output inference path connected to the High Value observation triggers the event. The operator has a specified period of time in which to acknowledge that the alarm for the Event Window Gate has passed a value of .true. The example shows the result when the alarm has not been acknowledged in time, so that the Event Window Gate passes a value of .false.



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
The Recurring Condition Gate	page 387	*Reference Manual*
The Remote Queue Message Block	page 400	*Reference Manual*
The Sound Bite Block	page 415	*Reference Manual*
The Local Alarm Panel	page 485	*Reference Manual*
The Group Alarm Panel	page 495	*Reference Manual*
The Alarm Readout	page 498	*Reference Manual*

Explanation Memory

The Explanation Memory capability remembers the explanation that the attached condition had when its input status value last matched the attribute Trigger On. The attribute Display Explanation On specifies the queue to which the explanation is added. The attribute Display When Updated specifies whether to display the explanation automatically whenever the attached block receives the status value Trigger On. This capability is useful when you want an explanation for a specific status value, but the status value may change before you can choose current explanation.

The attribute Trigger On can be .true, .false, or unknown. To remember explanations for more than one status value, attach more than one Explanation Memory capability to the block.

This capability adds one of these menu choices to the attached block's menu, depending on the value of Trigger On: explanation of last true, explanation of last false, or explanation of last unknown. When you choose one of these choices, the capability copies its stored explanation to the queue specified in the attribute Display Explanation On and displays that queue. Note that the explanation is copied to the queue only after you choose one of the menu choices.

If you want the capability to add the explanation to the queue and display the queue automatically when the attached block receives the status value Trigger On, set the attribute Display When Updated to yes. Otherwise, set it to no.

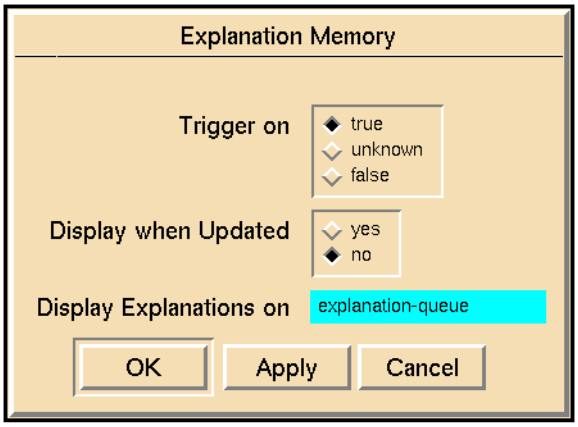
For information about the blocks to which you can attach an Explanation Memory capability, see "Specifying and Generating Explanations" on page 93.

Remembering Why an Alarm Was Raised

To remember why an alarm was raised even after it is lowered, attach both Alarm and Explanation Memory capabilities to the same block and set their Trigger On attributes to the same value. GDA notices that you want to remember the reason for the alarm, and adds the menu choice explain alarm in memory to the Alarm Panel.

Configuring

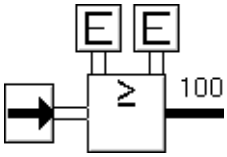
This is the configuration panel for the Explanation Memory capability.



Attribute	Description
Trigger On	The truth value of the block to which the capability is attached, whose explanation the capability remembers.
Display When Updated	Whether the capability displays the Explanation queue automatically when it detects the Trigger On value.
Display Explanations On	The name of the queue on which the capability displays the explanation message.

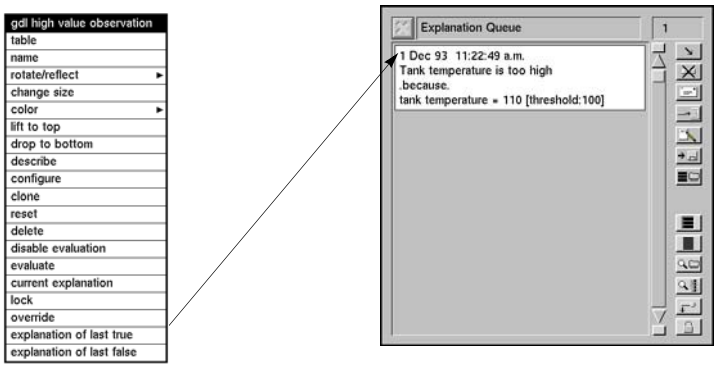
Example

The Explanation Memory Capabilities in the following example remember the explanations for the last .true and .false values. In one of the capabilities Trigger On is set to .true, and in the other Trigger On is set to .false.

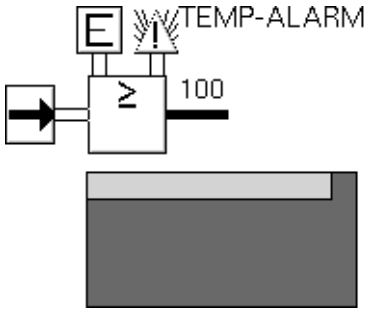


To see one of the saved explanations, select explanation of last true or explanation of last false from the menu of the High Value block, as shown in the next figure. Note that the choice explanation of last unknown is not in the block's menu, since

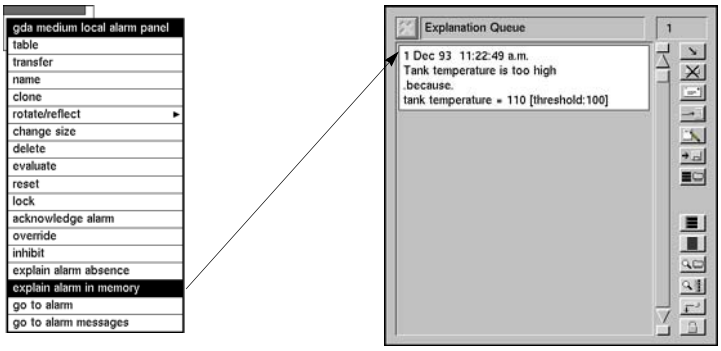
the block is not attached to an Explanation Memory capability which saves that information.



The Explanation Memory capability in this figure remembers the reason for the last alarm. The Trigger On attributes in both the Explanation Memory capability and the Alarm capability are set to .true. The Alarm Panel below the High Value block is for the alarm.



To see the saved explanation, you can select explanation of last true from the High Value block, or you can choose explain alarm in memory from the Alarm Panel's menu, as shown in this figure:



If you have more than one Explanation Memory capability attached to a block, the explain alarm in memory menu choice displays only the latest explanation created.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*
The Local Alarm Panel	page 485	*Reference Manual*
The Group Alarm Panel	page 495	*Reference Manual*
The Log Capability	page 537	*Reference Manual*
The Local Explanation Restriction	page 563	*Reference Manual*
The Ignore Path Explanation Restriction	page 565	*Reference Manual*

Log Capability

The Log Capability records changes in an inference port's value to a file. To specify the name of the log file, set the attribute Log Entries To. To specify what changes to log, set the attribute Log On to one of these values:

If Log On is…	Log Capability records whenever the inference port…
all-changes	Changes its value
.true	Becomes .true
.false	Becomes .false
unknown	Becomes unknown

When the Log Capability first evaluates, it opens a file with the name specified in Log Entries To. When you reset G2 or the capability, the capability closes the file. When you change the value of Log Entries To, the capability closes the file and opens a file with the new name.

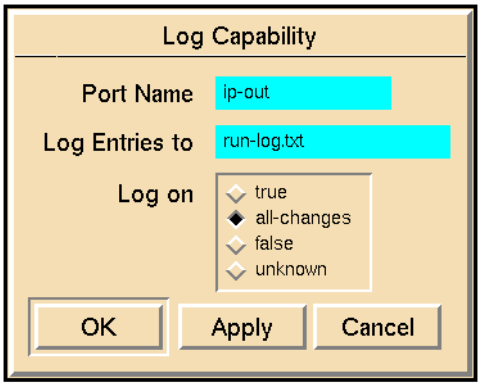
Each log entry contains the time and date of the change and a description of the change. If the belief value is not exactly 0.0, 0.5, or 1.0, the entry also gives the belief value.

You can attach a Log Capability to these blocks:

- Any block on the Observations palette.
- Any block on the Conditions palette.
- The Belief Entry Point block on the Entry Points palette.
- The Inference Output Action block on the Control Actions palette.
- The Network Belief Entry Point block on the Network palette.

Configuring

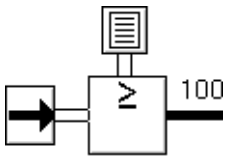
This is the configuration panel for the Log Capability .



Attribute	Description
Portname	The name of the inference output port whose changes the capability records in the log.
Log Entries To	The name of the log file in which to log the entries.
Log On	The truth value that cause the capability to create an entry in the log.

Example

The Log Capability in this figure logs whenever the temperature goes above or below 100. In the High Value Observation, Explanation When False is "Tank temperature is OK", Explanation When Unknown is "Tank temperature is becoming too high", and Explanation When True is "Tank temperature is too high"-



Here are some entries from the log file:

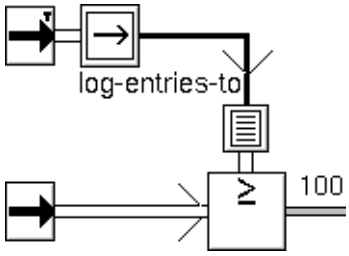
18 Aug 93 6:22:40 p.m.
Tank temperature is OK

18 Aug 93 7:28:07 p.m.
Tank temperature is becoming too high

18 Aug 93 8:28:14 p.m.
Tank temperature is too high

18 Aug 93 10:28:34 p.m.
Tank temperature is OK with belief 0.05

Suppose you want the Log Capability to keep each day's records in a different file. In this figure, a Text Entry Point connected to a Set Attribute block changes the name of the log file every day. Every day, the Set Attribute block changes the value of the Log Entries To attribute, so the Log Capability closes the old log file and opens a new log file with the new name.



To have the Text Entry Point generate a new file name every day, edit the subtable for its dp-out attribute. Set the Default Update Interval to 1 day and the Formula to an expression that generates a unique file name. In the Set Attribute block, set Target Attribute to Log Entries To. This table lists the settings for the Text Entry Point's subtable attributes:

Attribute	Value
Formula	"T1-[the current month]-[the current day of the month]-[the current year].log"
Default Update Interval	1 day

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*

For more information on...	See...	In this book...
The Explanation Memory Capability	page 533	*Reference Manual*
The Local Explanation Restriction	page 563	*Reference Manual*

Chart Capability


A Chart Capability is a link between the attribute of a block you want to display and the chart on which it is displayed. The Capability lets you choose where and how the attribute's data is displayed. In addition, it adds a menu choice to the chart that lets you modify the chart.

You can attach a Chart capability to any block that has a named data or inference input or output path, for example, dp-out or ip-in.

Setting Up a Chart

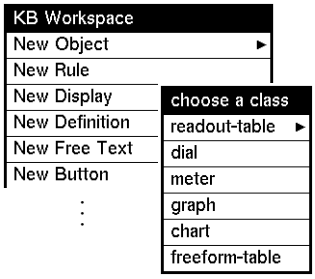
To set up a chart to display data from a block:

1. Create a new chart using KB Workspace > New Display. This attaches a new display to your mouse.
2. Position the display on the workspace and click with the mouse to place it.
3. Select configure from the chart's menu to display the configuration panel for the display.
4. Configure the chart and select the OK button to update the display.

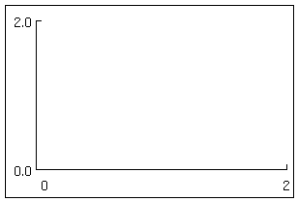
   GDA creates a Chart Capability with a system-defined chart name and places it next to the display.
5. Connect the Chart Capability to the block whose values you want to plot.
6. Select configure from the capability's menu to display its configuration panel. Configure the capability and select the OK button. For more information, see "Configuring a Chart" on page 543.

The following figure illustrates these steps for a chart:

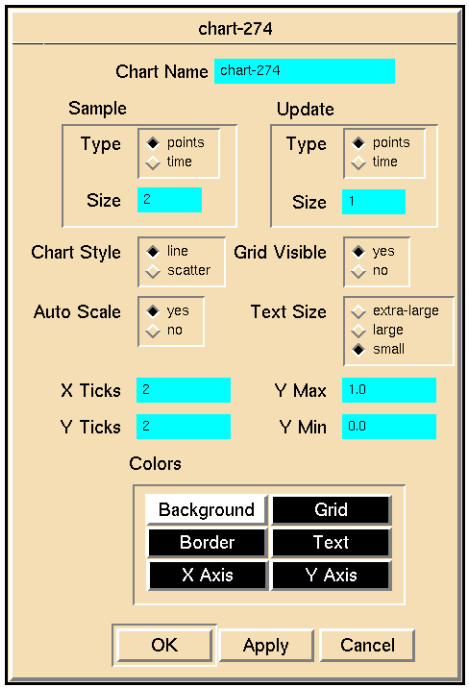
1.



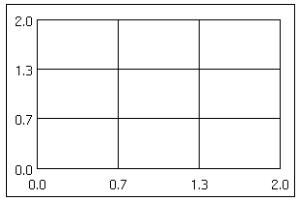
2.



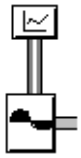
3.



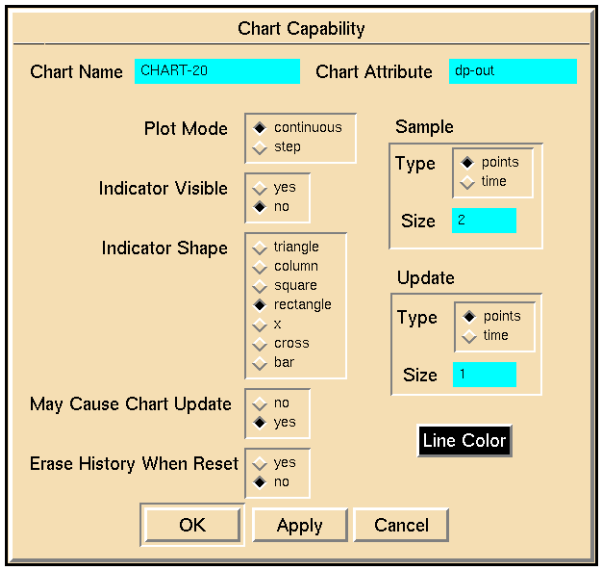
4.



5.



6.

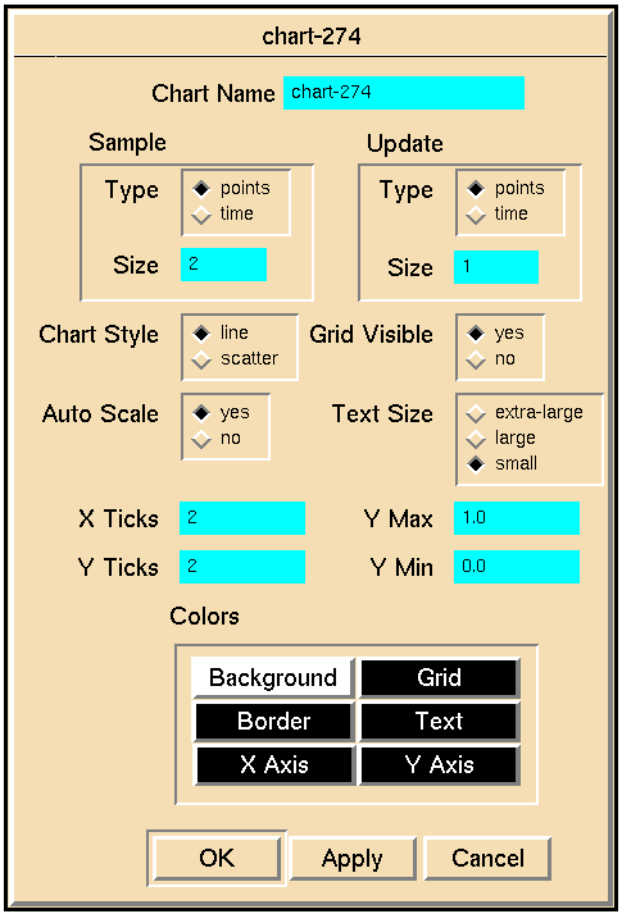


Caution When there is a single capability associated with a particular chart, GDA prevents you from configuring the chart and its associated capability at the same time. When there are multiple capabilities that plot data on a single chart, however, you should avoid simultaneous editing of the display and its capability, because GDA cannot prevent this.

Configuring a Chart

You configure the chart associated with its capability by using a configuration panel. The panel lets you set how much data the chart displays, how frequently the chart is updated, and what its axes look like.

To display the configuration panel for a chart, select configure from the chart's menu as shown:



Setting the Amount of Data Displayed

The field Sample Type specifies how points are displayed on the chart and how the field Sample Size is interpreted.

If Sample Type is…	Data points are displayed…	And Sample Size is…
Points	At equal intervals, regardless of when they arrived.	Number of points to display.
Time	According to the time they arrived.	Time interval to display.

For example, if Sample Type is Points and Sample Size is 60, the chart displays the last 60 points. If Sample Type is Time and Sample Size is 60 seconds, the chart displays the points received in the 60 second interval before the block last evaluated.

Setting How Frequently a Chart is Updated

When a block attached to a Chart Capability receives a value, the Chart Capability checks the setting of the attribute May Cause Chart Update. If that attribute is yes, the capability signals the chart to let it know it has received a new value. The chart then decides whether to update itself by checking the settings of Update Type and Update Size. If Update Type is Points, the chart is updated if it has received a total of Update Size signals since the last update. If Update Type is Time, the chart is updated if Update Size seconds have passed since the last update.

If Update Type is…	The chart is updated when a capability signals it and …
Points	The chart has received a total of Update Size signals since the last update.
Time	Update Size seconds have passed since the last update.

Note Updating a chart can consume a significant amount of your computer's time.

Specifying the Axes

To set how many tickmarks are on the chart's axes, edit X Ticks and Y Ticks. To display the tickmarks on the chart, set Grid Visible to true. To turn off the tickmarks, set Grid Visible to false.

To set the upper and lower limits for the Y axis, edit Y Max and Y Min, and set Auto Scale to no. To cause the chart to set its own upper and lower limits, set Auto Scale to yes; the chart ignores the values of Y Max and Y Min.

Determining How the Plot is Displayed

To chart values using a line between plot points, set Chart Style to line.

To chart values using points only, with no line between points, set Chart Style to scatter. When using a scatter-type plot, you must also set Indicator Visible to yes in the Chart Capability's configuration panel as shown in "Choosing How a Block's Data is Displayed" on page 545.

Setting the Chart Colors

You can set the following colors for a chart:

The color labeled...	Sets...
Background	The background color of the entire chart
Grid	The line color of the grid
Border	The line color of the border
Text	The text color of all text in the chart
X Axis	The line color of the x axis
Y Axis	The line color of the y axis

To set a color:

1. Click on the color you want to set.
2. Choose a color from the list of colors in the scroll area.
3. Select the OK button in the scroll area.
4. Select the Apply or OK button in the configuration panel.

Choosing How a Block's Data is Displayed

By default, a Chart Capability displays a block's data as a black line on the chart. To change how a Chart Capability displays its data on its chart, edit the capability's attribute table.

Specifying the Name of the Chart

The name of the chart must correspond to the Chart Name specified in the capability. GDA automatically inserts a system-generated name in the configuration panel for the Chart Capability when you configure the associated chart; thus, there is no need to update these attributes in the capability unless you want to change the name. For a description, see step 4 under "Setting Up a Chart" on page 541.

You can also enter an expression that evaluates to the name of a chart. For example, you can use an expression such as "[chart-variable]" to refer to a variable whose value is the name of a chart, or you can use an expression such as "[(the chart-name of chart-object)]" to refer to an attribute of an object whose value is the name of a chart.

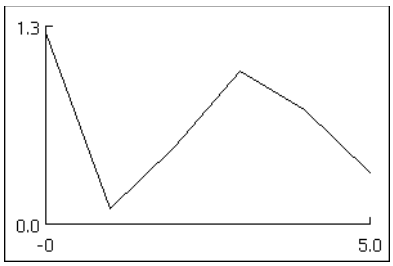
Specifying The Attribute to Plot

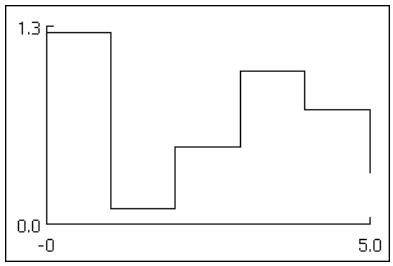
The attribute Chart Attribute determines which attribute is plotted in the chart. The default value of this attribute is dp-out, which plots the data output value of the block to which the capability is connected. For example, when plotting entry points and signal generators, you do not need to edit this attribute.

Specifying the Type of Connection Between Points

To choose how the display connects the points in a block's data set, set the attribute Plot Mode to one of the values in the following table. The figure shows you what a chart that uses these modes looks like.

If Plot Mode is…	Then the display…
continuous	Directly connects the points in the block's dataset.
discrete	Inserts extra points into the block's dataset so the transitions are square.





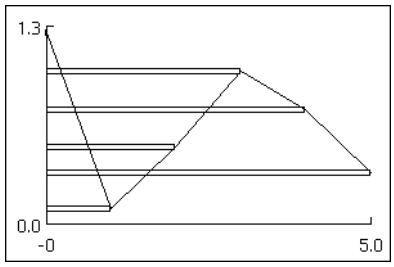
This attribute is ignored by charts when Chart Style is scatter in the Chart Capability configuration panel as discussed in "Determining How the Plot is Displayed" on page 545.

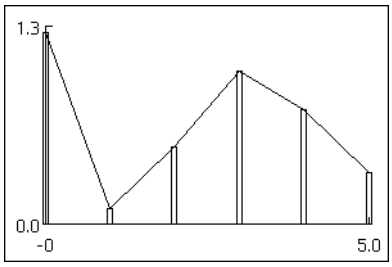
Specifying a Marker

In a Chart Capability, you can mark each point in a block's data set. Set the attribute Indicator Visible to yes. To choose a marker, set the attribute Indicator Shape to one of these values:

Attribute	Picture
rectangle	□
square	■
triangle	△
cross	+
x	×
bar	see below
column	see below

The bar and column indicators are a little different from the rest. They draw a line from the point to an axis. A bar goes to the Y axis, and a column goes to the X axis, as shown in this figure:





Specifying Whether a Capability Can Update a Display

In a Chart Capability, you can choose whether getting a new value can force the chart to update. If the attribute May Cause Chart Update is yes, the capability signals the chart every time it receives a new value, and the chart may update if the time is right. (You determine when the right time is by setting the fields Update Type and Update Size in the chart's configure dialog, described in "Setting How Frequently a Chart is Updated" on page 544.) If May Cause Chart Update is no, the capability does not let the chart know it has received a new value.

If several Chart Capabilities use a single chart, it is good practice to let only the one furthest downstream update the chart. This practice lessens the time GDA spends updating charts and makes sure that the displayed values are consistent.

Specifying the Line Color

To specify the color of the line used to plot data in a chart:

1. Click on Line Color in the configuration panel for the Chart Capability.
2. Choose a color from the color palette.
3. Select the OK button in the scroll area.
4. Select the Apply or OK button in the configuration panel.

Going to a Chart

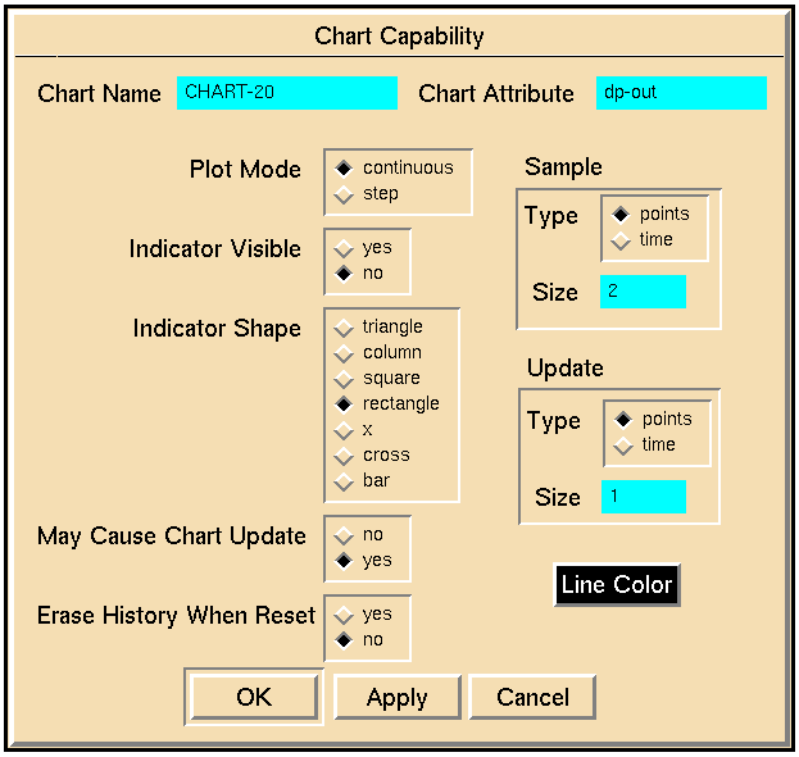
Sometimes you need to put a chart on a different workspace from a Chart Capability. To see the display for a capability, choose go to chart from the capability's menu. GDA brings the workspace containing the display to the front.

Resetting

What happens when you reset a Chart Capability depends on the setting of the attribute Erase History When Reset. If Erase History When Reset is yes, the capability deletes its history of values. The next time it plots a point, it erases the line before plotting the new point.

Configuring

This is the configuration panel for the Chart Capability.

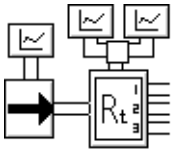


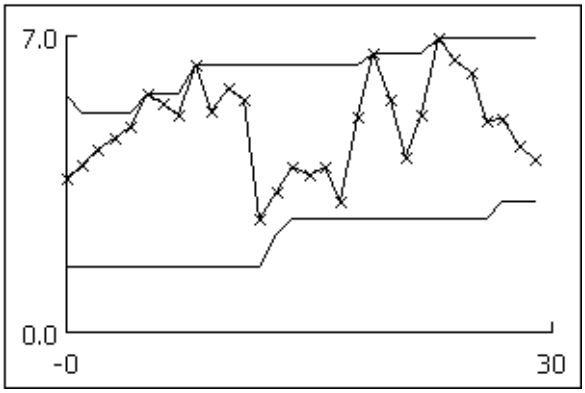
Attribute	Description
Chart Name	The name of the chart on which the capability plots its data.
Chart Attribute	The attribute of the block to which the capability is attached whose data will plot.
Plot Mode	Whether the capability connects the data points to form a continuous plot, or whether the capability inserts additional points to create a step-like plot.
Indicator Visible	Whether the data points are visible or not.
Indicator Shape	When Indicator Visible is yes, the shape of the indicators that are the data points of the plot.

Attribute	Description
May Cause Chart Updating	Whether the chart automatically updates each time it receives a new data value or not.
Erase History When Reset	See “Specifying What Happens to History Upon Reset” on page 90 in the *GDA User’s Guide*.
Sample Type and Sample Size	See “Specifying the Size of the History” on page 85 in the *GDA User’s Guide*.
Update Type and Update Size	See “Specifying When to Propagate Data” on page 88 in the *GDA User’s Guide*.
Line Color	The color of the plot.

Examples

The Chart Capabilities in the following figure display the values from an Entry Point, in addition to the minimum and maximum values from the entry point for the past fifteen points. The Entry Point’s values are marked with an X.





The following table lists some of the attribute values for the Chart Capabilities. Notice that there are two capabilities attached to the Moving Range block: one charts the maximum and one charts the minimum.

Attribute Name	Entry Point	Maximum	Minimum
Chart Name	chart-1	chart-1	chart-1
Chart Attribute	dp-out	dp-out-1	dp-out-3
Line Color	black	black	black
Indicator	x	rectangle	rectangle
Indicator Visible	yes	no	no
Plot Mode	continuous	continuous	continuous
May Cause Chart update	no	no	yes
Erase History When Reset	no	no	no

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*

Clock

The Clock capability forces the attached block to evaluate at a specified interval and not necessarily when the block receives new input data. You can attach a Clock capability to any block.

To specify the interval, set the attribute Evaluation Period. For example, if Evaluation Period is 1 minute, the block will evaluate once every minute.

If you do not want the block to evaluate when it receives new data, set Allow Intermediate Evaluation to no. The block will evaluate only at the specified interval.

To turn the Clock capability on and off, use the menu choices enable evaluation and disable evaluation.

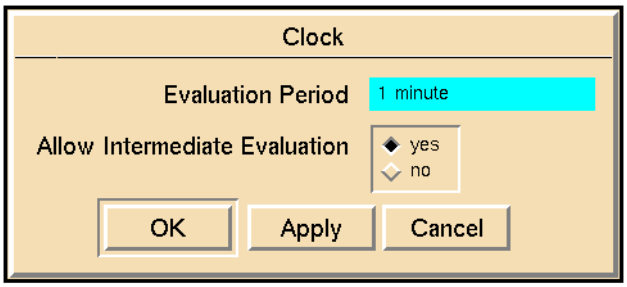
When you turn the clock off, the block to which the Clock capability is attached evaluates whenever it receives a new value, as opposed to every Evaluation Period. When you turn the clock on, the attached block evaluates once every Evaluation Period.

If the Clock capability has Allow Intermediate Evaluation set to no, the attached block does not fire when the clock is turned off.

Note You can attach a Clock capability to a Log Capability to cause the attached capability to update at regular intervals.

Configuring

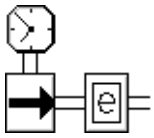
This is the configuration panel for the Clock capability.



Attribute	Description
Evaluation Period	The time interval at which the clock evaluates.
Allow Intermediate Evaluation	Whether the clock also evaluates when it receives new data (yes) or only every Evaluation Period (no).

Example

The Clock capability in this figure forces the Data Entry Point to pass a value at a regular interval, even if its value does not change:



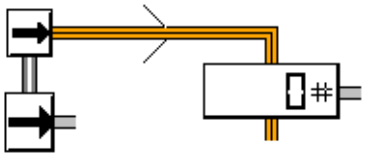
See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Control Initiation Capability

The Control Initiation Capability initiates a control action whenever its associated block receives a value. You connect the capability to any block via the link, and you connect the control path to any control action. For example, you can use a Control Initiation Capability to increment a counter each time a block evaluates, as shown in the following diagram:



Configuring

The Control Initiation Capability has no configurable attributes.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Dialog Restriction

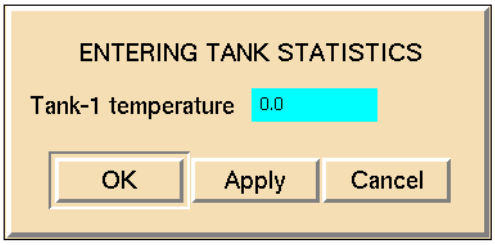
The Dialog Restriction lets you change the prompts in an Override dialog. You can attach a Dialog Restriction to any block that supports the override menu choice. For more information, see “Overriding Block Values” on page 51

Customizing a Numeric or Text Entry Point Override Dialog

When the Dialog Restriction is attached to a Numeric Entry Point or a Text Entry Point, the attribute Override Text specifies the text on the top of the dialog, and the attribute New Value Prompt specifies the text below the edit field. Generally, the Override Text describes why the data is needed and the New Value Prompt describes the information needed.

The attributes Override Text and New Value Prompt can include the values of certain G2 expressions, as described in “Evaluating Expressions in Attributes” on page 118 in the *GDA User's Guide*. For example, the string "temperature for [the contents of tank-1] tank" might appear as "temperature for water tank", where water is the contents of tank-1.

For example, a Numeric Entry Point with a Dialog Restriction attached created the following dialog. In the Dialog Restriction, Override Text is "ENTERING TANK STATISTICS" and New Value Prompt is "Tank-1 temperature".



Customizing a Belief Entry Point or Condition Override Dialog

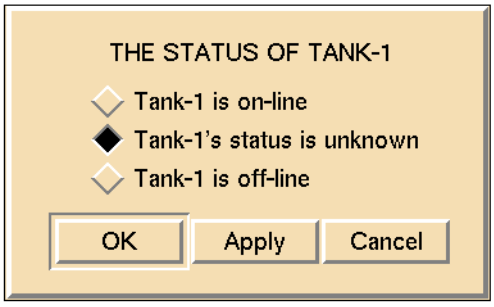
When the Dialog Restriction is attached to a Belief Entry Point or a Condition, the attribute Override Text specifies the text at the top of the dialog. The restriction does not use the attribute New Value Prompt.

Note A Condition is any block in the palettes Conditions and Observations, except for the High and Low block.

To specify the text for the three radio buttons in the Override dialog, you do not need a Dialog Restriction. Instead, set the attributes Description When True, Description When Unknown, and Description When False in the block. The block will use those attributes in the Override dialog even if there is no Dialog Restriction attached.

Note The attributes Override Text, Description When True, Description When Unknown, and Description When False can include the values of certain G2 expressions, as described in "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*. For example, the string "status for [the contents of tank-1] tank" might appear as "status for water tank", where water is the contents of tank-1.

For example a Belief Entry Point with a Dialog Restriction attached created the following dialog. In the Dialog Restriction, Override Text is "THE STATUS OF TANK 1". In the Belief Entry Point, Description When True is "Tank 1 is on-line", Description When Unknown is "Tank 1's status is unknown", and Description When False is "Tank 1 is off-line".



Specifying the G2 Windows in which the Dialog Appears

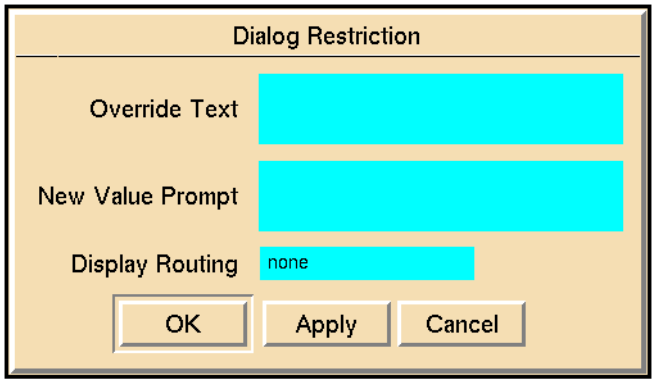
To specify which G2 windows the dialog should appear in, use the Display Routing attribute. It can have these values:

If Display Routing is...	The dialog appears in these G2 windows...
A Display Routing object	All the windows specified in the object
A G2 Window	That G2 window
none	All G2 windows

Note Display Routing is an object on the Network palette in the Other Tools menu. It lets you give a name to a group of G2 windows. A G2 window can be a Telewindows connection to a G2 process or a G2 process.

Configuring

This is the configuration panel for the Dialog Restriction.



Attribute	Description
Override Text	The text at the top of the override dialog.

Attribute	Description
New Value Prompt	The text below the dialog when creating a dialog for a Numeric or Text Entry Point.
Display Routing	The G2 window on which the dialog appears.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Overriding Block Values	page 51	*User's Guide*
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*

Manual Entry Restriction

The Manual Entry Restriction indicates that the attached Entry Point gets its value only from the operator with a dialog, and not from an external data source, G2 simulator, or G2 inference engine. This restriction also lets you change the text in the dialog, which looks much like an Override dialog.

There are two situations in which you might use Manual Entry Restrictions:

- If you have an alarm in a diagram and the diagram's entry points are not receiving data, you may want to manually enter data for the entry points to see if there is cause for an alarm.
- If you are using an encapsulation block, you might want to enter values for the entry points on its subworkspace before the encapsulation block executes.

Setting the Dialog's Appearance and Position

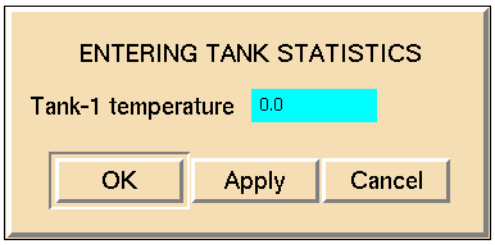
This section explains how to customize the prompts in the Manual Entry Restriction's dialog and how to choose where the dialog appears.

Customizing a Numeric Entry Point or Text Entry Point Override Dialog

When the Manual Entry Restriction is attached to a Numeric Entry Point or a Text Entry Point, the attribute Override Text specifies the text on the top of the dialog, and the attribute New Value Prompt specifies the text below the edit field. Generally, the Override Text describes why the data is needed and the New Value Prompt describes the information needed.

Note The attributes Override Text and New Value Prompt can include the values of certain G2 expressions, as described in "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*. For example, the string "temperature for [the contents of tank-1] tank" might appear as "temperature for water tank", where water is the contents of tank-1.

For example a Numeric Entry Point with a Manual Entry Restriction attached created the following dialog. In the Manual Entry Restriction, Override Text is "ENTERING TANK STATISTICS" and New Value Prompt is "Tank-1 temperature".



Customizing a Belief Entry Point or Condition Override Dialog

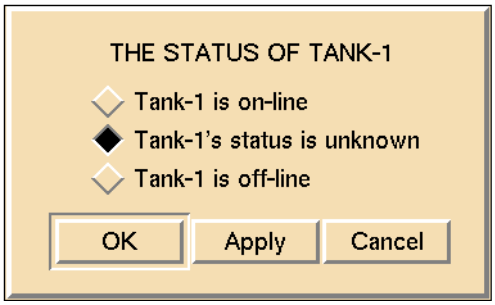
When the Manual Entry Restriction is attached to a Belief Entry Point or a Condition, the attribute Override Text specifies the text at the top of the dialog. The restriction does not use the attribute New Value Prompt.

Note A Condition is any block in the palettes Conditions and Observations, except for the High and Low block.

To specify the text for the three radio buttons in the Override dialog, you do not need a Manual Entry Restriction. Instead, set the attributes Description When True, Description When Unknown, and Description When False in the block. The block will use those attributes in the Override dialog even if there is no Manual Restriction attached.

Note The attributes Override Text, Description When True, Description When Unknown, and Description When False can include the values of certain G2 expressions, as described in "Evaluating Expressions in Attributes" on page 118 in the *GDA User's Guide*. For example, the string "status for [the contents of tank-1] tank" might appear as "status for water tank", where water is the contents of tank-1.

For example a Belief Entry Point with a Manual Entry Restriction attached created the following dialog. In the Manual Entry Restriction, Override Text is "THE STATUS OF TANK 1". In the Belief Entry Point, Description When True is "Tank 1 is on-line", Description When Unknown is "Tank 1's status is unknown", and Description When False is "Tank 1 is off-line".



Specifying the G2 Windows in which the Dialog Appears

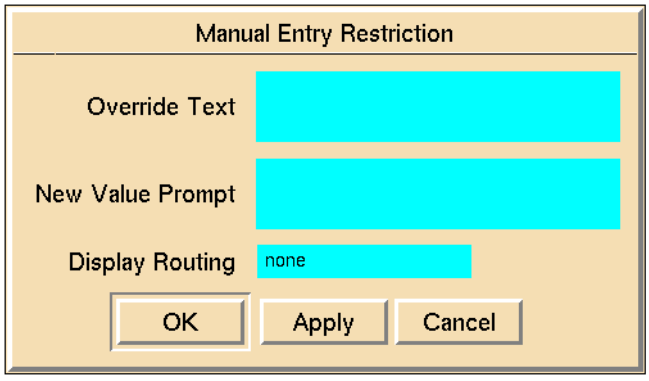
To specify which G2 windows the dialog should appear in, use the Display Routing attribute. It can have these values:

If Display Routing is...	The dialog appears in these G2 windows...
A Display Routing object	All the windows specified in the object
A G2 Window	That G2 window
none	All G2 windows

Note Display Routing is an object on the Network palette in the Other Tools menu. It lets you give a name to a group of G2 windows. A G2 window can be a Telewindows connection to a G2 process or a G2 process.

Configuring

This is the configuration panel for the Manual Entry Restriction.



Attribute	Description
Override Text	The text at the top of the override dialog.
New Value Prompt	The text below the dialog when creating a dialog for a Numeric or Text Entry Point.
Display Routing	The G2 window on which the dialog appears.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*
Evaluating Expressions in Attributes	page 118	*User's Guide*
The Dialog Restriction	page 555	*Reference Manual*

Local Explanation Restriction

The Local Explanation Restriction indicates that the attached condition provides the explanation for this path. GDA stops searching further upstream and uses the explanation that the attached block produces.

You can attach a Local Explanation Restriction to these blocks:

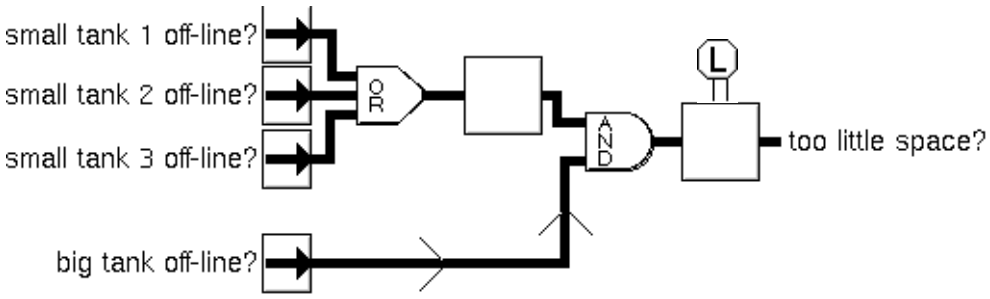
- Any block on the Observations palette.
- The Conclusion block and the Inference Query block on the Conditions palette.
- The Inference Output Action block on the Control Actions palette.
- The Network Belief Entry Point block on the Network palette.

Configuring

The Local Explanation Restriction has no configurable attributes.

Example

The Local Explanation Restriction in the following figure lets a Conclusion provide an explanation for the three entry points that are upstream from it. This diagram determines whether there is too little tank space available. If any small tank is off-line and the big tank is off-line, there is not enough tank space.



With the Local Explanation Restriction, choosing current explanation from the last Conclusion's menu produces this explanation:

```
Not enough space is available
.because.
(The big tank is off-line
        .and.
At least one small tank is off-line)
```

If you did not use the Local Explanation Restriction, current explanation produces this explanation:

```
Not enough space is available
.because.
The big tank is off-line
        .and.
(Tank 1 is off-line
        .or.
Tank 2 is off-line
        .or.
Tank 3 is off-line)
```

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*
The Local Alarm Panel	page 485	*Reference Manual*
The Group Alarm Panel	page 495	*Reference Manual*
The Alarm and Recurring Alarm Capabilities	page 519	*Reference Manual*
The Log Capability	page 537	*Reference Manual*
The Ignore Path Explanation Restriction	page 565	*Reference Manual*

Ignore Path Explanation Restriction

The Ignore Path Explanation Restriction removes a path from an explanation. The path that contains the block attached to this restriction does not supply a clause to the explanation.

It lets you filter out irrelevant paths when producing an explanation. For example, an ignored path might provide production statistics that do not contribute to the alarm you want explained.

You can attach an Ignore Path Explanation Restriction to these blocks:

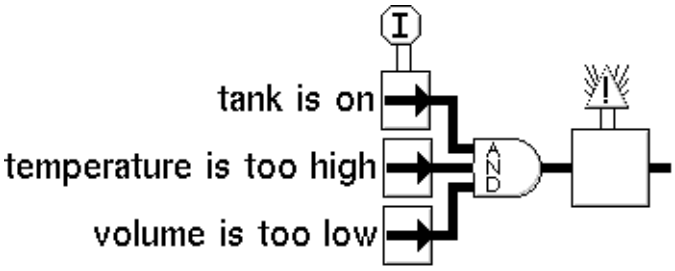
- Any block on the Observations palette.
- The Conclusion block and the Inference Query block on the Conditions palette.
- The Inference Output Action block on the Control Actions palette.
- The Network Belief Entry Point block on the Network palette.

Configuring

The Ignore Path Explanation Restriction has no configurable attributes.

Example

The Ignore Path Explanation Restriction in the following figure lets you ignore the explanation from an entry point. This diagram concludes that a tank is in danger if the tank is on, its temperature is too high, and its volume is too low. Since the operator knows that the tank can be in danger only if the tank is on, that reason is left out of the explanation.



With the Ignore Path Explanation Restriction, choosing current explanation from the last Conclusion's menu produces this explanation:

```
Tank 1 is in danger
.because.
(Temperature is too high
        .and.
Volume is too low)
```

If you did not use the Ignore Path Explanation Restriction, current explanation produces this explanation:

```
Tank 1 is in danger
.because.
(Tank is on
        .and.
Temperature is too high
        .and.
Volume is too low)
```

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*
The Local Alarm Panel	page 485	*Reference Manual*
The Group Alarm Panel	page 495	*Reference Manual*
The Alarm and Recurring Alarm Capabilities	page 519	*Reference Manual*
The Log Capability	page 537	*Reference Manual*
The Local Explanation Restriction	page 563	*Reference Manual*

24

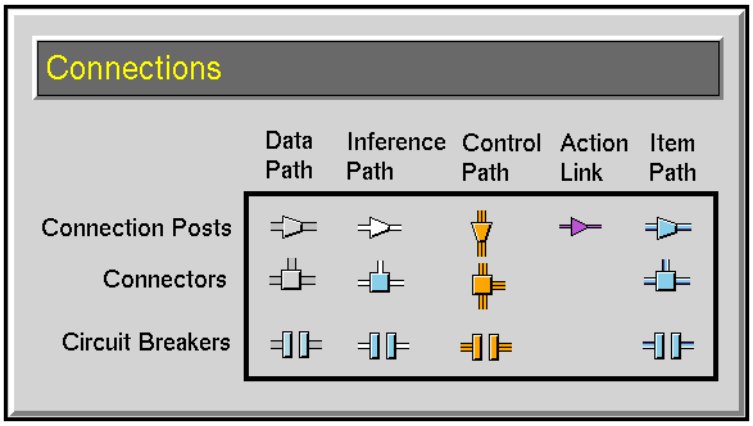
Connections

Describes the objects that control how information flows through paths.

Introduction

The Connections palette contains connections that control how information flows through paths.

You find the Connections palette on the Other submenu of the Palettes menu:



These objects do not modify data or perform an action, but they let you control how information flows through paths. You can let paths cross workspaces, join paths together, and create loops.

The Connection Posts, Connectors, and Circuit Breakers associated with Item Paths behave the same way as the connections associated with the other types of paths, except they pass items. For more information, see "Connection Posts" on page 569, "Connectors" on page 572, and"Circuit Breakers" on page 574.

For more information on item paths, see "Using Item Paths" on page 63 in the *GDA User's Guide*, see "Stub Tools" on page 615, "Customizing the Connection Stubs" on page 175 in the *GDA User's Guide*, and the *GDA API Reference*.

Connection Posts

The "Connection Posts" on page 569 let paths cross workspace boundaries so you can create diagrams that span several workspaces.

Connectors

The "Connectors" on page 572 allow you to join paths with a branch-in topology. They pass the last value received.

Controlling Feedback Cycles

GDA allows you to create diagrams containing loops. You use circuit breakers to prevent GDA from entering an infinite loop.

The "Circuit Breakers" on page 574 let you control loops in your diagram.

See Also

The Entry Points on page 7 lets you enter data into your GDA diagram.

Connection Posts

Connection Posts let paths cross workspace boundaries so you can create diagrams that span several workspaces. G2 passes the data from an output connection post to all input connection posts with the same name. GDA has five types of Connection Posts: Data Path Connection Posts, Inference Path Connection Posts, Control Path Connection Posts, Action Link Connection Posts, and Item Path Connection Posts. They are identical except for the type of data they handle.

A Connection Post must have a name to work. An output connection post and its corresponding input connection posts must have the same name to pass data properly. To name the connection post, configure the connection post.

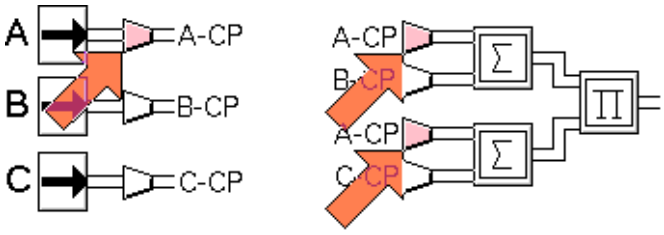
The Connection Posts on the Connections palette can function as input or output. To change one to an input post, just delete the input stub by dragging it onto the post, and vice versa for an output post.

Connection Posts let you break up a complex diagram and put related parts of the diagram on different workspaces. You can also use input and output connection posts on the same workspace to reduce the number of crossed paths.

Note Do not modify the Superior-connection attribute of a connection post.

Highlighting

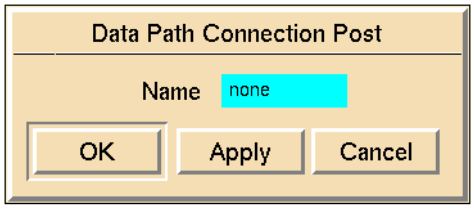
If your diagram is large and contains many Connection Posts, you may want to see the output Connection Post that sends data to another post, or you may want to find all the input posts that receive data from another post. To highlight all connection posts with the same name, choose highlight from a Connection Post's menu. GDA brings to the front all workspaces that contain a connection post with that name, colors the post the block highlight color (pink, by default), and places an arrow beside it.



To change the post's color back and remove the arrow, choose do not highlight from the post menu for any of the highlighted posts.

Configuring

This is the configuration panel for a Data Path Connection Post:



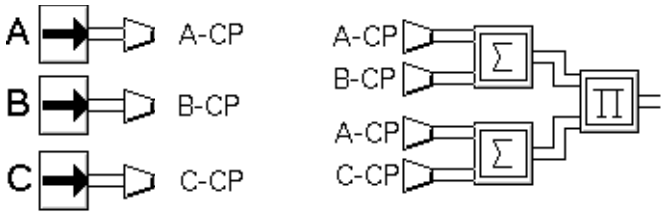
Attribute	Description
Name	The name of the connection post.

Examples

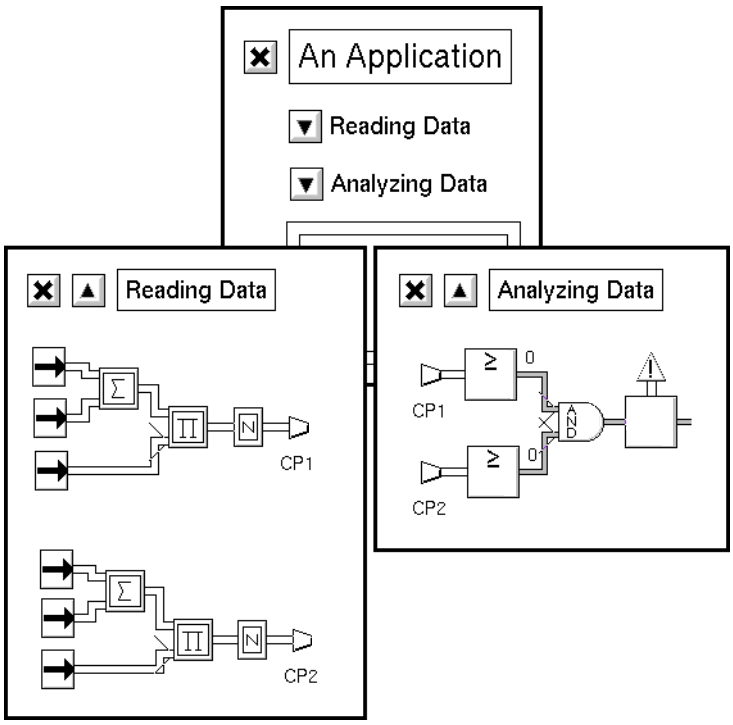
The following figure shows the equation:

$(a + b)(a + c)$

The Connection Posts let you use the value from Entry Point A in two different places, without crossing paths.



The Connection Posts in the following figure carry data from the workspace named "Reading Data" to the workspace named "Analyzing Data." They let you split the application into two pieces, so you can work on it more easily.

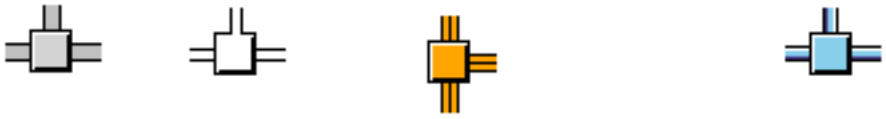


See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Using Paths	page 60	*User's Guide*

Connectors



A Connector lets you combine two output paths, so a block's input port can get a value from either path. A connector passes the last value received on any of its input paths. GDA has four types: Data Path Connectors, Inference Path Connectors, Control Path Connectors, and Item Path Connectors.

The various types of Connectors are nearly identical, except for the type of data they handle.

Connectors are similar to splitters, which split an output path into two paths, so that one output path can enter two ports. G2 creates splitters for you automatically whenever you connect an input path to another path.

You can drag additional paths into a connector, similar to a peer input block.

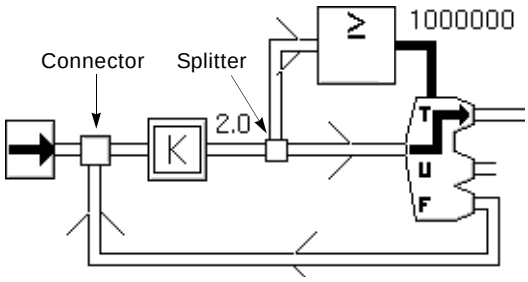
Note G2 does not create Connectors automatically. You must create them yourself by cloning them from the Connections palette. If you try to connect an output path to another path without a Connector, G2 deletes the path and generates a warning message.

Configuring

The Connectors have no configurable attributes.

Example

The diagram in the following figure doubles the value in a data path until the value exceeds one million (1,000,000). A Connector combines the output paths from an Entry Point and a Data Switch, so the Gain block can get input from either. Also note that a Splitter splits the output from the Gain block into two paths, so both a High Value block and the Data Switch can use it.



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*

Circuit Breakers

Circuit Breakers allow you to control a GDA diagram with a loop. They are not required if another block can halt the iteration of the loop. Otherwise, you use the circuit breaker to prevent infinite loops in a diagram.

When the Circuit Breaker is open, the value on the path is propagated across the breaker unchanged, but GDA does not evaluate the downstream block immediately. The next block evaluates when it receives data from somewhere else. When the Circuit Breaker is closed, the value on the path is propagated across the breaker, and the downstream block evaluates immediately.

Circuit Breakers contain a menu choice for opening and closing the circuit breaker. If the circuit breaker is open, the menu choice is close breaker, and if the circuit breaker is closed, the menu choice is open breaker.

Opening and closing the circuit breaker also changes the icon as shown in the following figure:

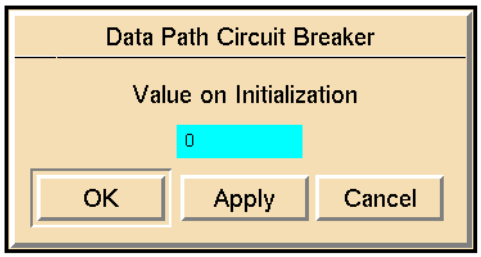
‘All types of circuit breakers are open by default.

Configuring

Circuit Breaker configuration panels vary depending on the type of circuit breaker.

Configuring the Data Path Circuit Breaker

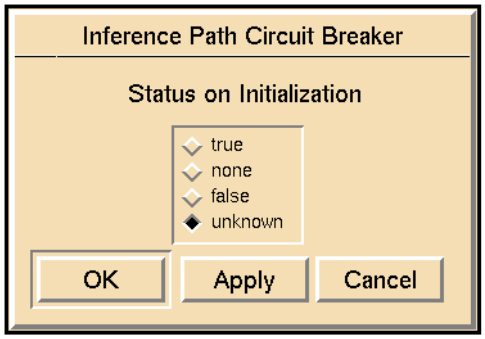
This is the configuration panel for the Data Path Circuit Breaker.



Attribute	Description
Value on Initialization	See “Specifying an Initial Data Value” on page 83 in the *GDA User’s Guide*.

Configuring the Inference Path Circuit Breaker

This is the configuration panel for the Inference Path Circuit Breaker.



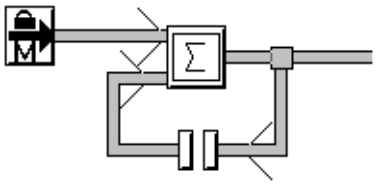
Attribute	Description
Status on Initialization	See “Specifying an Initial Status Value” on page 84 in the *GDA User’s Guide*.

Configuring the Control Path and Item Path Circuit Breakers

The Control Path Circuit Breaker and the Item Path Circuit Breaker have no configurable attributes.

Example

In the following example, the Circuit Breaker is open, which allows the Summation block to accumulate values from the Numeric Entry Point each time a new value appears on the path. Do not close the Circuit Breaker in this situation.



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Understanding the GDA Block Evaluation Engine	page 126	*User's Guide*

25

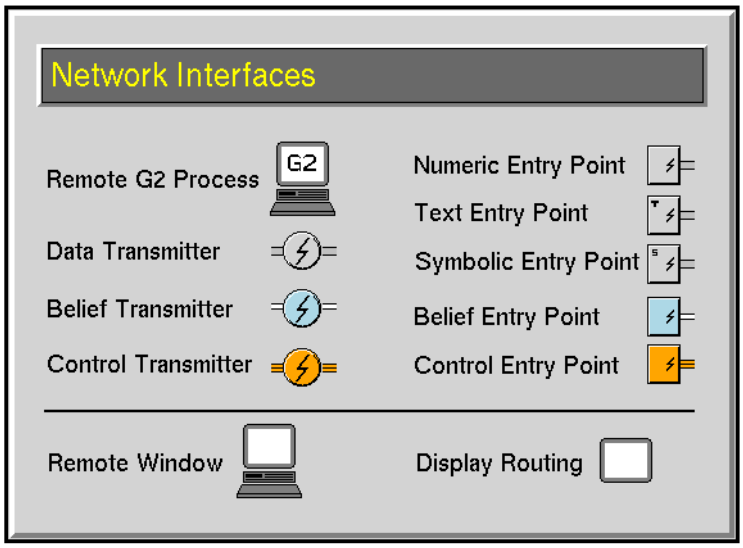
Network Interfaces

Describes the blocks that let you send information from one G2 to another.

Introduction

The objects on this palette let you send information from one G2 application to another and let you create groups of Telewindows connections that will display a dialog.

You can find the Network Interfaces palette in the Other submenu of the Palettes menu:



Sending Information from One GDA Application to Another

These blocks let you send information from a GDA application on one machine to a GDA application on another machine over a computer network:

- The Remote G2 Process block on page 580 specifies the GDA application that will receive the information.
- The Transmitter blocks on page 585 send information to another application.
- The Network Entry Point blocks on page 589 receive information from another GDA application.

Specifying Which G2 Windows Display a Dialog

These objects let you specify which G2 windows can display a dialog:

- The Remote Window block on page 592 specifies one Telewindows connection.
- The Display Routing block on page 594 contains an array of Remote Window objects and other Display Routing objects.

See Also

These blocks perform functions similar to the blocks in this palette:

- Entry Points on page 7 let you enter data into your GDA diagram.
- The Data Output block on page 145 lets you send data from a GDA diagram to a G2 variable or parameter.
- The Connection Posts on page 569 let you pass information from one workspace to another, within the same G2 process.

Remote G2 Process

The Remote G2 Process object maintains a connection to a G2 process running on another computer on the network. When you create a Transmitter to send information to another G2 process, you set the Transmitter's G2 Process attribute to the name of a Remote G2 Process object.

Caution If you delete a Remote G2 Process block that is connected to a remote G2 process, G2 sometimes aborts.

Specifying the Address

To specify the address of the remote G2 process, configure these attributes:

- **Host**: the name of the machine running the remote G2. If the Remote G2 Process object cannot determine the machine's name, this is UNKNOWN.
- **Port Number/DECnet Object**: the port number of the remote G2. To find the port number, go to the remote G2 and choose Main Menu > Miscellany > Network Information.
- **Network Type**: the type of network connecting the machines. It can be either TCP/IP or DECnet.

Reading the Status of the Remote Process

The Remote G2 Process object updates its display to reflect the current status of the connection. The color of the icon and the display text changes to reflect the status. The possible values and their default colors are listed in the following table:

If the Remote G2 Process is...	Its color is…	And…
connected	blue	This object is establishing a connection to the remote G2.
not-connected	gray	The object has not tried to establish a connection.

If the Remote G2 Process is...	Its color is…	And…
timed-out	red	The object timed-out while trying to establish a connection.
error	yellow	The object failed to establish a connection.
running	green	The remote G2 is running.
reset	orange	The remote G2 is reset.
paused	magenta	The remote G2 is paused.

To change these default colors, go to Preferences > Colors > Networks.

Setting What the Status Message Looks Like

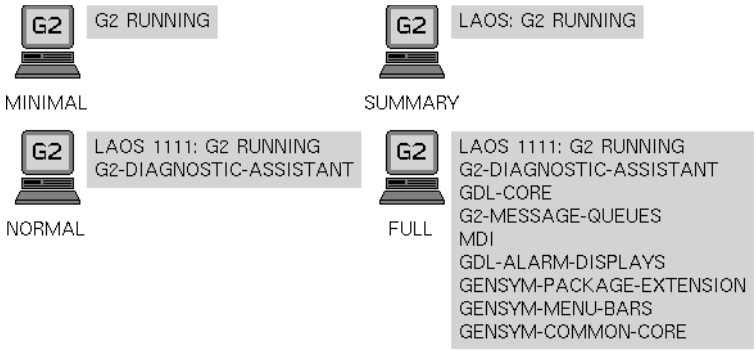
The Remote G2 Process object displays status information about its connection beside its icon. Several attributes in the Remote G2 Process object let you control how this information is displayed, and one menu choice in its menu lets you update the display.

Specifying the Contents of the Display

To control how much information the Remote G2 Process icon displays, set the attribute Display Format to one of the values in the following table:

If Display Format is…	The display contains the…
minimal	Attribute Inference Status
full	Attributes Host, Port Number, and Inference Status
	Names of the all packages loaded in the process
summary	Attributes Host and Inference Status
normal	Attributes Host, Port Number, and Inference Status
	Name of the primary package loaded in the process

This figure shows what these display formats look like:



Specifying Where the Display Appears

The status information can appear to the left or right side of the icon. To choose one, set the attribute Display Location to one of these values:

If Display Location is…	The display appears on the…
rhs	Right side of the Remote G2 Process icon
lhs	Left side of the Remote G2 Process icon

Specifying the Size of the Display

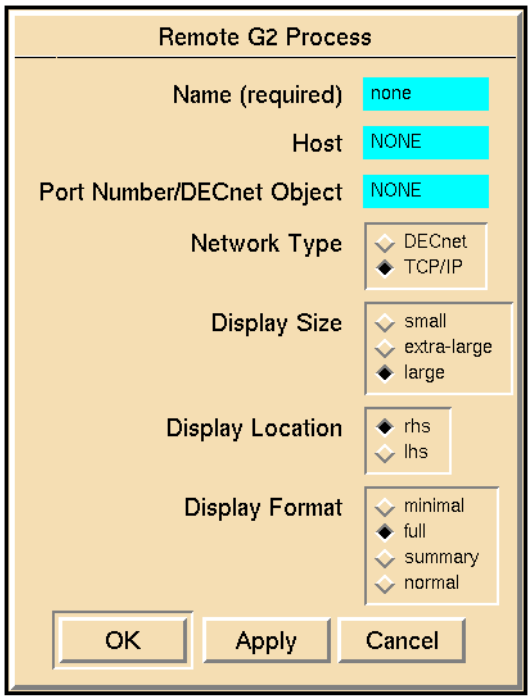
To set the size of the font that the status information is printed with, set the attribute Display Size to small, large, or extra-large. These are the standard G2 font sizes.

Initializing the Display

Usually, the Remote G2 Process object updates its display only when it notices that the process has changed. If you want to update the display immediately, choose initialize from the item's menu.

Configuring

This is the configuration panel for the Remote G2 Process object.



Attribute	Description
Name	The name of the Remote G2 Process block that the Transmitter block references.
Host	The name of the host machine running the remote G2.
Port Number/DECnet Object	The port number of the remote G2, or the name of the DECnet object for the remote G2.
Network Type	The type of network connecting the machines.
Display Size	The font size of the display information.

Attribute	Description
Display Location	Whether the display information appears to the right (rhs) or the left (lhs) of the icon.
Display Format	How much information the Remote G2 Process icon displays.

See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
The Transmitter blocks	page 585	*Reference Manual*
The Network Entry Points	page 589	*Reference Manual*

Transmitters

The Transmitter block sends information to a GDA application running on another machine on the network.

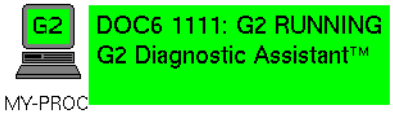
To receive the information in the other application, use a Network Entry Point block on page 589. The Data Transmitter block can send information to a Numeric, String, or Symbolic Network Entry Point. The Belief Transmitter can send information to a Belief Network Entry. And a Control Transmitter can send information to a Control Network Entry Point.

Setting up a Link between Two GDA Applications

To send information from one GDA application to another:

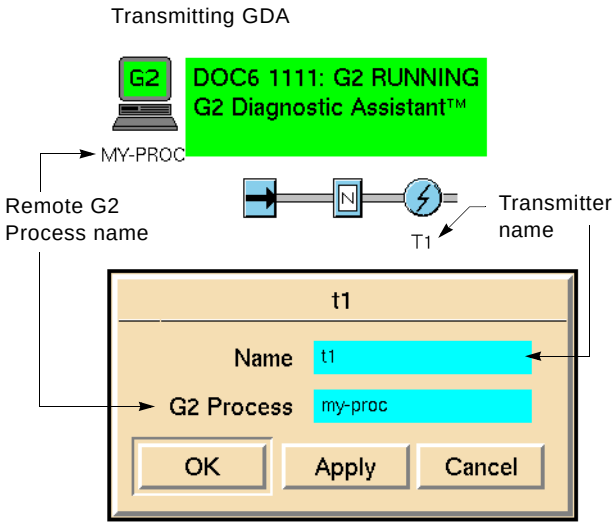
1 Set up a Remote G2 Process object that specifies the GDA application that will receive the information. Be sure to configure the object's Host, Port Number, and Network Type attributes, and to specify the Name attribute.

The following figure shows a Remote G2 Process object that is set up for a machine on a TCP/IP network:



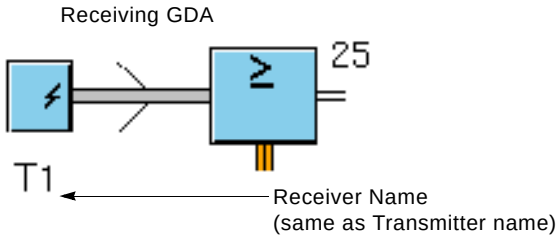
2 Create a transmitter in the GDA application that sends the information. Set the attribute G2 Process to the Name attribute of the Remote G2 Process. Give the block a name by configuring the Name attribute.

This figure shows an application that filters data from an entry point, and then sends it to another application with a Data Transmitter:



3 Create a Network Entry Point in the GDA application that receives the information. Specify the Name of the entry point to be the same as the name of the transmitter.

This figure shows an application that receives data from another application with a Numeric Network Entry Point:

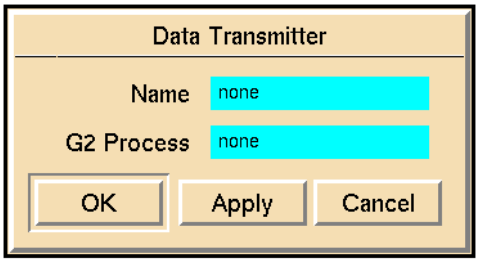


Configuring

Transmitter block configuration panels vary depending on the type of the block.

Configuring the Data Transmitter and Control Transmitter

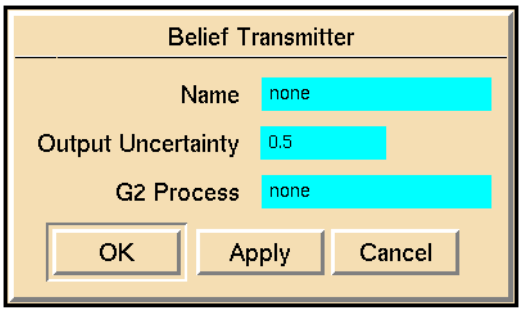
This is the configuration panel for the Data Transmitter. The panel for the Control Transmitter is identical except for the block name.



Attribute	Description
Name	The name of the transmitter block that the Network Entry Point references.
G2 Process	The name of the Remote G2 Process block that connects to the remote G2.

Configuring the Belief Transmitter

This is the configuration panel for the Belief Transmitter.



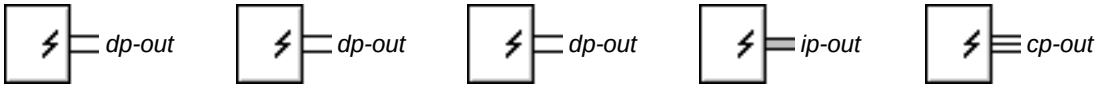
Attribute	Description
Name	The name of the transmitter block that the Network Entry Point references.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
G2 Process	The name of the Remote G2 Process block that connects to the remote G2.

See Also

For more information on how to use these blocks, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*
Entry Points	page 7	*Reference Manual*
Data Output	page 145	*Reference Manual*
Connection Posts	page 569	*Reference Manual*
Network Entry Points	page 589	*Reference Manual*

Network Entry Points



The Network Entry Points receive data from a GDA application running on another machine on the network.

To send information from a GDA application, use a Transmitter block on page 585. The Numeric, Text, and Symbolic Network Entry Points receive data from a Data Transmitter. The Belief Network Entry Point receives data from a Belief Transmitter. The Control Network Entry Point receives data from a Control Transmitter.

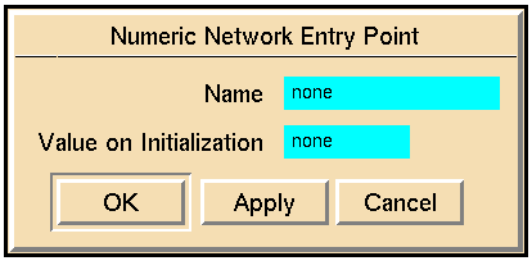
The Belief Network Entry Point contains the attribute Transmitting Host. GDA fills this attribute with the name of the host that is sending information to the Entry Point, and it uses that name to create an explanation for this Entry Point. (The name does appear in the explanation, however.) This attribute is read-only.

Configuring

Network Entry Point configuration panels vary depending on the type of the entry point.

Configuring the Numeric, Text, Symbolic, and Control Network Entry Points

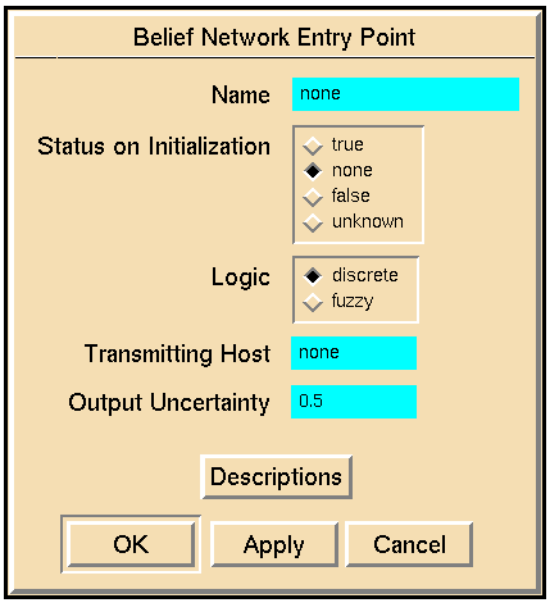
This is the configuration panel for the Numeric Network Entry Point. The panels for the Text, Symbolic, and Control Network Entry Points are identical except for the block names.



Attribute	Description
Name	The name of the Network Entry Point, which refers to the name of the transmitter that is sending the data.
Value on Initialization	See "Specifying an Initial Data Value" on page 83 in the *GDA User's Guide*.

Configuring the Belief Network Entry Point

This is the configuration panel for the for the Belief Network Entry Point.



Attribute	Description
Name	The name of the Network Entry Point, which refers to the name of the transmitter that is sending the data.
Status on Initialization	See "Specifying an Initial Status Value" on page 84 in the *GDA User's Guide*.
Logic	See "Specifying the Type of Logic to Use" on page 101 in the *GDA User's Guide*.

Attribute	Description
Transmitting Host	(Read-only) The name of the host that is sending data to the Entry Point, which the Entry Point uses when generating explanations.
Output Uncertainty	See "Specifying Uncertainty" on page 102 in the *GDA User's Guide*.
Description when True, Description when False, Description when Unknown	See "Specifying an Explanation" on page 94 in the *GDA User's Guide*.

See Also

For more information on how to use all these blocks, see the sections below.

For more information on...	See...	In this book...
Basic Block Behavior	page 48	*User's Guide*
Overriding Block Values	page 51	*User's Guide*
Specifying Initial Values	page 83	*User's Guide*
Entry Points	page 7	*Reference Manual*
Connection Posts	page 569	*Reference Manual*

For more information on how to use the Belief Network Entry Point, see the sections below.

For more information on...	See...	In this book...
Specifying and Generating Explanations	page 93	*User's Guide*
Specifying the Type of Logic to Use	page 101	*User's Guide*
Specifying Uncertainty	page 102	*User's Guide*

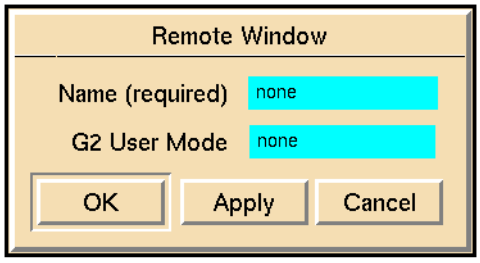
Remote Window

The Remote Window object specifies a G2 window, which is a Telewindows connection to a G2 process or the G2 process. The Display Routing object contains a list of Remote Window objects to let you specify which Telewindows connections should display a particular dialog.

For more information on how to set up a G2 window, see the *Telewindows User's Manual*.

Configuring

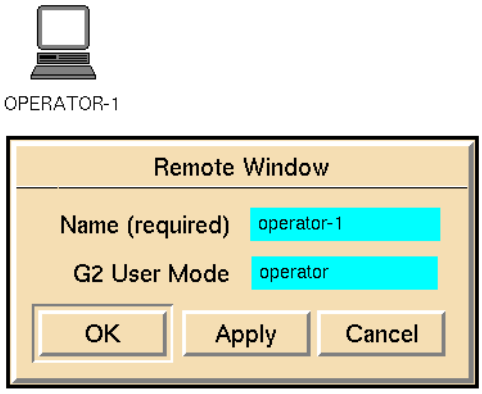
This is the configuration panel for the Remote Window object.



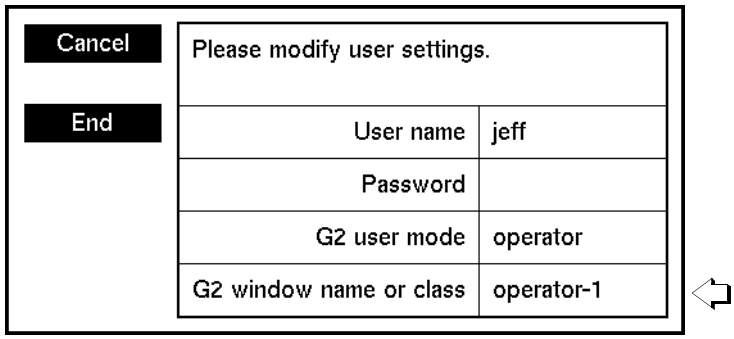
Attribute	Description
Name	The name of the Remote Window object, which the Display Routing object references.
G2 User Mode	The user mode in which the remote window operates.

Example

The Remote Window object in the following figure specifies a G2 window with a Name of operator-1 and a G2 User Mode of operator.



This figure shows a Change Mode dialog that specifies the Remote Window object, OPERATOR-1, created above:



See Also

For more information on how to use this block, see the sections below.

For more information on...	See...	In this book...
The Remote Window object	page 592	*Reference Manual*

Display Routing

The Display Routing object lets you give a name to a group of G2 windows. A G2 window can be a Telewindows connection to a G2 process or the G2 process. It lets you specify which Telewindows connections should display a particular dialog.

Specifying the Windows

To create a Display Routing object:

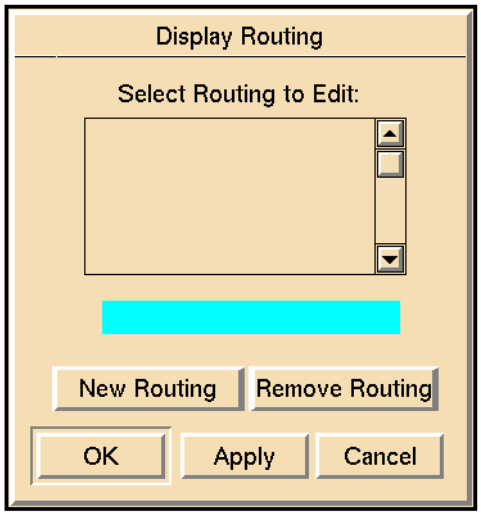
1. Create a number of Remote Window objects, which will be part of the Display Routing object.
2. Configure the Display Routing object by choosing the configure menu choice.
3. Click the New Routing button to create a new routing object.
4. Enter the name of an existing Remote Window object, g2-window object, or Display Routing object.

   GDA confirms that the name you specify is a valid G2 window or Display Routing object.

   The window name appears in the scroll area.
5. Create as many new routing objects as desired to define the group of windows that make up the Display Routing object, and click OK.

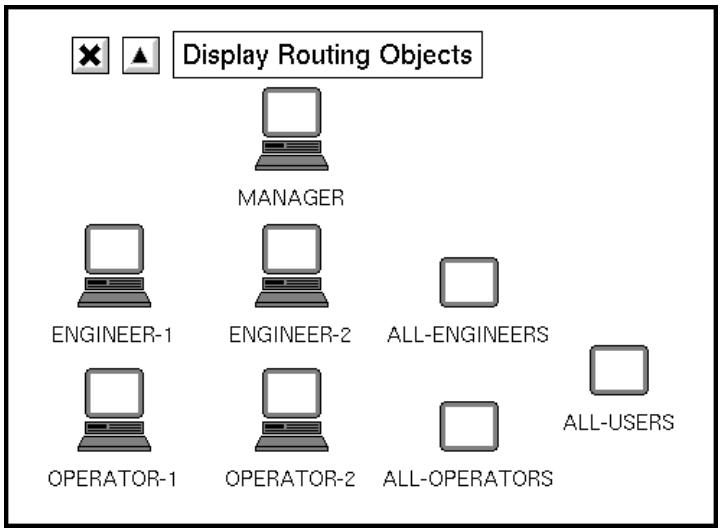
Configuring

This is the configuration panel for the Display Routing object.



Example

The following figure shows a typical arrangement of G2 windows. This application uses three Display Routing objects: one for all the engineer's windows, another for all the plant operators, and another for all the users of the GDA application.



This table lists the Remote Window objects for three Display Routing objects:

Display Routing Objects:	ALL-ENGINEERS	ALL-OPERATORS	ALL-USERS
Remote Window Objects:	ENGINEER-1 ENGINEER-2	OPERATOR-1 OPERATOR-2	ALL-ENGINEERS ALL-OPERATORS MANAGER

If the Display Routing attribute for a User Query block is all-operators, the User Query block's dialog appears only on the operators' windows. If the Display Routing attribute for the Error Queue is all-engineers, the Error Queue appears only on the Engineers' windows.

See Also

For more information on how to use this block, see the sections below.

For more information on...	**See...**	**In this book...**
The Inference Query block	page 381	*Reference Manual*
The User Query Control Switches	page 456	*Reference Manual*
The Show Subworkspace block	page 407	*Reference Manual*
The User Query blocks	page 469	*Reference Manual*
The Dialog Restriction block	page 555	*Reference Manual*

26

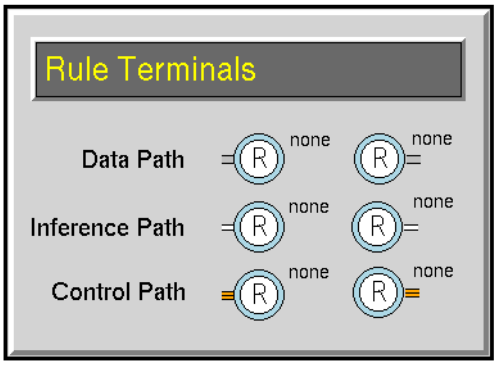
Rule Terminals

Describes the blocks you use to invoke G2 rules and conclude values.

Introduction

The blocks on the Rule Terminals palette invoke G2 rules and conclude values in a GDA diagram.

You find the Rule Terminals palette in the Other submenu of the Palettes menu:



There are two general categories of rule terminals:

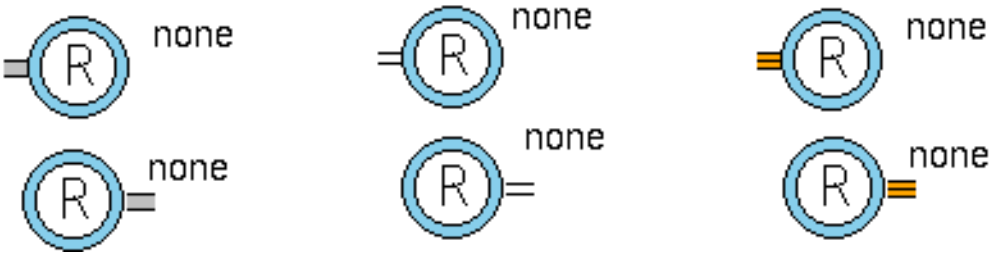
- **Invocation rule terminals**, which have input connection paths. When an invocation rule terminal receives a value, it invokes any rule that refers to that rule terminal.

- **Conclusion rule terminals**, which have output connection paths. When a rule concludes a value for a conclusion rule terminal, it passes the concluded value.

There are three types of rule terminals, corresponding to the three types of paths (GDA does not support item path rule terminals). The following table outlines each type and the values they receive and conclude:

Rule terminals of type...	Receive...	And conclude...
Data Path	Quantitative values (excluding text or symbols)	Quantitative values (excluding text or symbols)
Inference Path	Symbolic status values of .true, .false, or unknown	Symbolic status values of .true, .false, or unknown
Control Path	Control signals, which are integer values	A single control signal, which is an integer value

Invocation and Conclusion Rule Terminals



Rule terminals are G2 variables, which invoke rules and conclude values using forward chaining. Rule terminals are like any other G2 variable, in that they can keep histories and track expiration times. For a description of forward chaining and variables, see the *G2 Reference Manual*.

The following table describes when GDA evaluates each category of rule terminal and the effect this has on the rule and the rule terminal.

A rule terminal of category...	Evaluates when...	Which...
Invocation	The rule terminal receives a new value	Invokes all rules that refer to the rule terminal in the antecedent of the rule
Conclusion	A rule concludes a value for the rule terminal in the rule's consequent	Passes the value through the rule terminal

Specifying the Name Tag of the Rule Terminal

When an invocation rule terminal receives an input, the rule terminal invokes a rule, which can in turn conclude a value into a conclusion rule terminal.

The link between the rule terminal and the rule is the **name tag** of the rule terminal, which is the name to which the rule refers. You specify the name tag in one of two places:

- In the Name Tag attribute display when you initially clone the rule terminal. Click on none and enter a unique symbol.
- In the Name Tag attribute in the configuration panel for the rule terminal. Select configure from the rule terminal's menu and enter a unique symbol.

When you create a name tag for a rule terminal, the rule terminal displays the Name Tag as an attribute display:

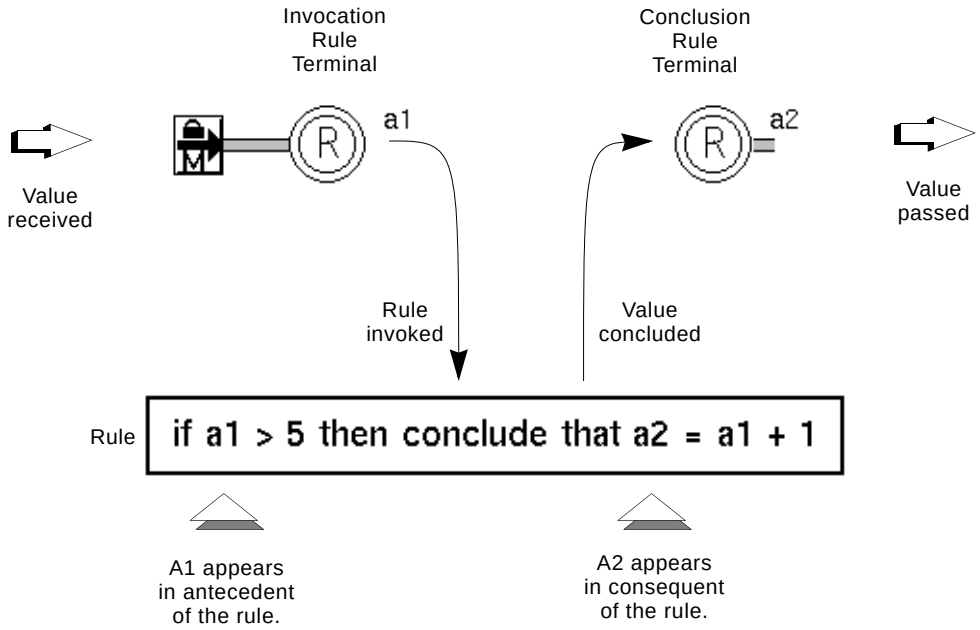
Note Do not use a G2 reserved word for the Name Tag; otherwise, errors will occur.

Referring to Rule Terminals in a Rule

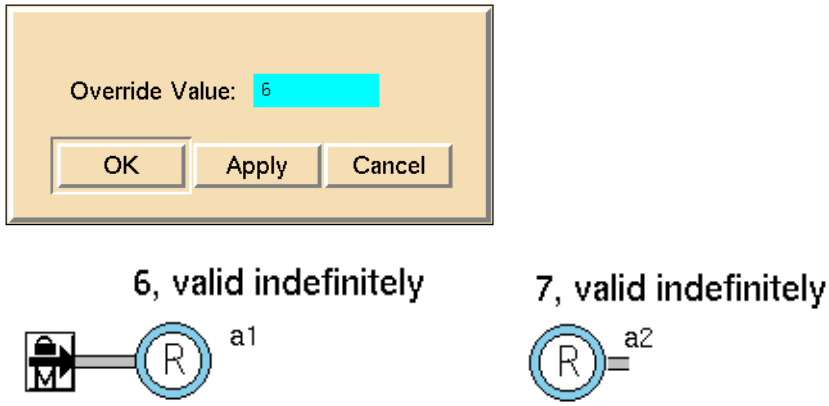
The simplest use of rule terminals is to create a single rule that refers to both invocation and conclusion rule terminals. Depending on the kind of rule terminal, you refer to the name tag of the rule terminal in either the antecedent or the consequent of the rule, as outlined in this table:

You refer to...	In the...
Invocation rule terminals	Antecedent of a rule
Conclusion rule terminals	Consequent of a rule

The following figure illustrates how you use a single rule both to invoke rules and conclude values. The rule tests whether the data value received by the invocation rule terminal (A1) is above 5, and concludes a value into the conclusion rule terminal (A2) that is one greater than the value of A1.



The following figure illustrates the result of passing a value of 6 into the invocation rule terminal. The figure shows the override dialog for the Numeric Entry Point and the current values of the Last Recorded Value attribute for each rule terminal.



Note You must use rule terminals as endpoints in a diagram; you cannot have a rule terminal with a path leading both into and out of the rule terminal.

Note When writing rules involving inference path values, you must always refer to the symbols .true and .false, as opposed to true and false.

Note Data must be enabled for conclusion rule terminals to work properly.

Caution Do not create an Invocation Rule Terminal with a rule that concludes a value into a Conclusion Rule Terminal, which has a rule that simultaneously concludes a value into the same Invocation Rule Terminal; otherwise, G2 enters a cyclical state, which requires that you restart G2.

Other Techniques of Using Rule Terminals

In the example above, the rule refers to a rule terminal in both the antecedent and the consequent of the rule, and the GDA diagram contains both an invocation and conclusion rule terminal. While this is the one common use of rule terminals, you can also use rule terminals to:

- Invoke rules that do not conclude values into other rule terminals, but rather, perform some action, such as informing the operator of a condition. See "Using Invocation Rule Terminals Alone on an SSE Diagram" on page 611 for an example.
- Start a control sequence based on a rule, which GDA invokes using external mechanisms. For example, you might have a set of G2 rules that detect a fire in a plant. If a fire is detected, a G2 rule concludes a value for a control path rule terminal, which initiates a GDA control sequence to shutdown the plant.
- Create "local variables" on a Single Source Encapsulation block using conclusion rule terminals, which both invoke rules and conclude values. For an example, see "Storing Local Values on an SSE Block Using Conclusion Rule Terminals" on page 610.

Using Rule Terminals on a Single Source Encapsulation Block

Because rule terminals do not receive and pass values like most blocks do, the most common use of rule terminals is in conjunction with a Single Source Encapsulation block. This allows you to hide the rules and rule terminals on a subworkspace. The end result is a block that essentially "passes values" based on a rule, where the user cannot see the rules and associated rule terminals in the diagram.

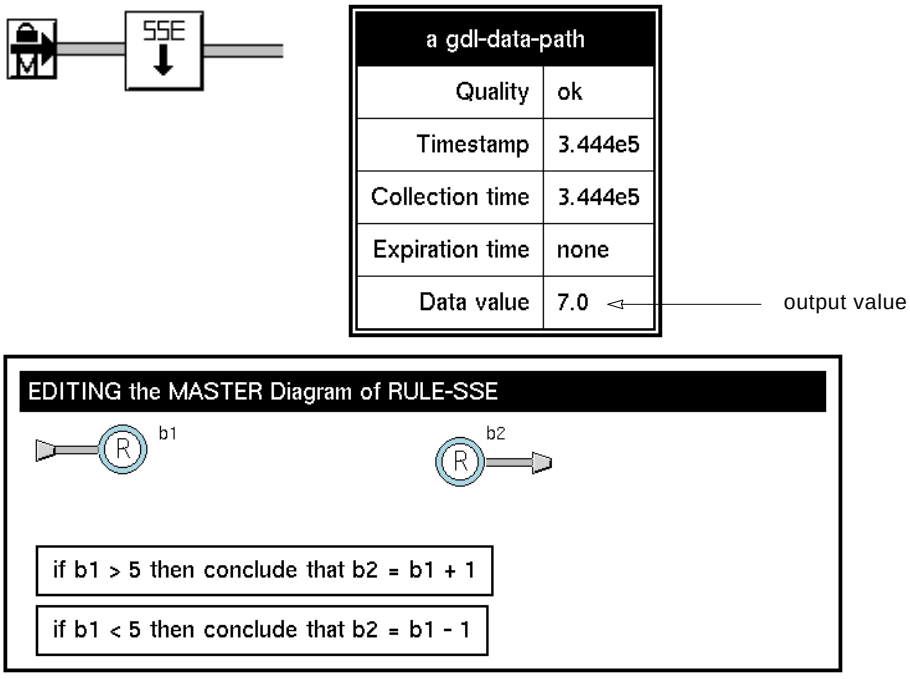
When you use rule terminals outside of an SSE block, there are no restrictions as to the type of rule you can invoke, i.e., you can invoke if rules, when rules, and whenever rules, unconditionally rules, for rules, and initially rules.

Note When you use rule terminals on a Single Source Encapsulation block, you can only invoke if and whenever rules.

The following figure shows the same example as shown at the beginning of this section except that the names are different and the rule and rule terminals are contained in a Single Source Encapsulation block.

The diagram shows the SSE block, the Master Diagram, and the output path value. The invocation rule terminal, b1, is connected to the entry connection post, and the conclusion rule terminal, b2, is connected to the exit connection post. When the SSE block receives a value, the value is passed to b1, which invokes the rule, which in turn concludes a value for b2, which is then passed through the SSE block.

The following figure shows the Master Diagram for the SSE block and the resulting output path value for an input value of 6:



Note If you have created multiple instances of a Single Source Encapsulation block that contains rule terminals, GDA only invokes the rules associated with the instance that received a value; it does not invoke rules associated with every instance of the particular class of SSE block.

Note If you use the Custom Class Wizard to create a subclass of an existing Single Source Encapsulation block that contains rules, the rules initially appear with ellipses. You must remove the ellipses from the rule by editing it before you save the Master Diagram to avoid errors when the block evaluates.

Note You cannot create scanned rules on an SSE, either by specifying a scan interval for the rule, or using a Clock capability. Rules are only invoked when the block evaluates.

Note Saving the Master Diagram locally using Apply Changes Locally does not apply changes made to rules on the Master Diagram. You must apply these changes globally.

Caution When including rules on a Single Source Encapsulation block, do not refer to Rule Terminals located outside of the encapsulation block's Master Diagram; otherwise, unpredictable results may occur.

For details on how to edit the Master Diagram of a Single Source Encapsulation, see "Single Source Encapsulation" on page 208 in the *GDA User's Guide*.

Saving a Master Diagram That Contains Rules

When editing a rule on a Master Diagram for a Single Source Encapsulation block, you must use the Save Diagram menu choice in the Master Diagram to save changes to the rule.

If you attempt to use the Apply Changes Locally menu choice, you receive a message indicating that you cannot apply changes to the rule locally. This means you cannot test changes to the rule locally for a single instance of a Single Source Encapsulation block; you must apply the changes to all instances of the class.

Note If a rule on a Master Diagram refers to a rule terminal that is not on the Master Diagram, or if the rule contains other inconsistencies, GDA displays an error. The background of the rule becomes yellow, the error queue indicates that there is a problem incorporating the rule into the diagram, and GDA displays a popup message allowing you to cancel or continue the save. Once you have fixed the error, select the reset menu choice to clear the error.

Displaying the Local Diagram of an SSE Block Using Rule Terminals

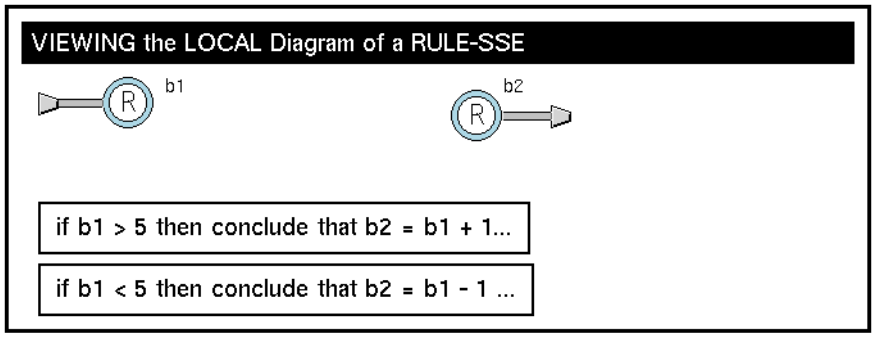
When you use view local diagram to display the Local Diagram of an SSE block containing rules, you notice three things:

- The rules contain ellipses (...) to indicate that you cannot edit this rule.
- The rules highlight when the block receives a value and invokes the rule.

Because the rules highlight when they are invoked, use the Local Diagram for an SSE block with rules when developing an application.

Note For performance reasons, GDA does not highlight rules outside of an SSE block. For debugging purposes, you can turn rule highlighting on for rules outside of SSE blocks by selecting Main Menu > Run Options > Highlight Invoked Rules.

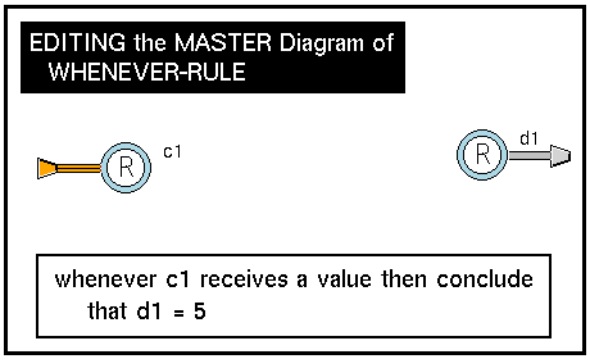
The following figure shows the Local Diagram for the example under "Using Rule Terminals on a Single Source Encapsulation Block" on page 602:



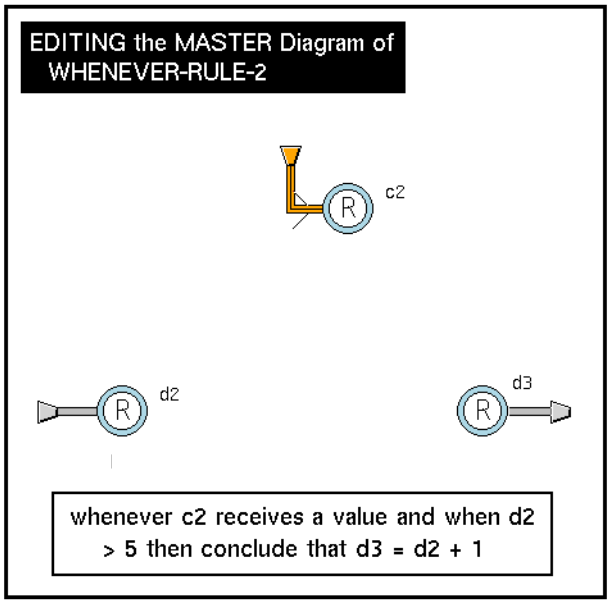
Specifying Whenever Rules on SSE Blocks

You can specify whenever rules on an SSE block, which are invoked by a rule terminal. Typically, you use whenever rules in conjunction with Control Path Rule Terminals to invoke a rule whenever the rule terminal receives a control signal.

For example, the following Control Path Rule Terminal named c1 invokes the whenever rule whenever c1 receives a control signal. The rule concludes a value for the Data Path Rule Terminal named d1.



GDA also supports the and when syntax in conjunction with whenever rules. In the following example, the rule is invoked whenever c2 receives a value and when d2 is greater than 5. The rule concludes a value for d3.



Note GDA does not support any other syntax for whenever rules for use with Rule Terminals.

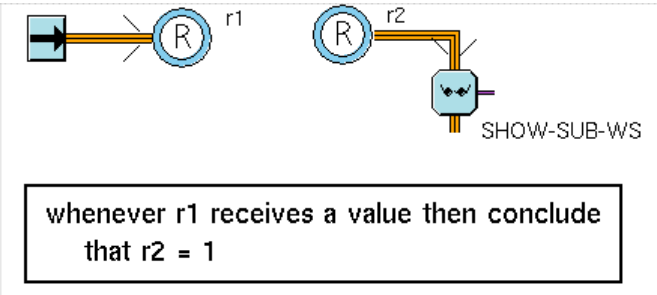
Using Control Path Rule Terminals

Control Path Invocation Rule Terminals send integer values on their output paths. Similarly, Control Path Conclusions Rule Terminals must receive integer values.

Typically, you associate a whenever rule with a Control Path Invocation Rule Terminal to invoke the rule when the rule terminal receives a control signal. The integer value increases by one with each control signal it receives. You can also use an if rule to test the value of the rule terminal's integer value.

To use a Control Path Conclusion Rule Terminal to send a control signal downstream, you must conclude an integer value into the rule terminal. You can conclude any integer value to send a control signal.

The following examples show how to use each type of Control Path Rule Terminal:



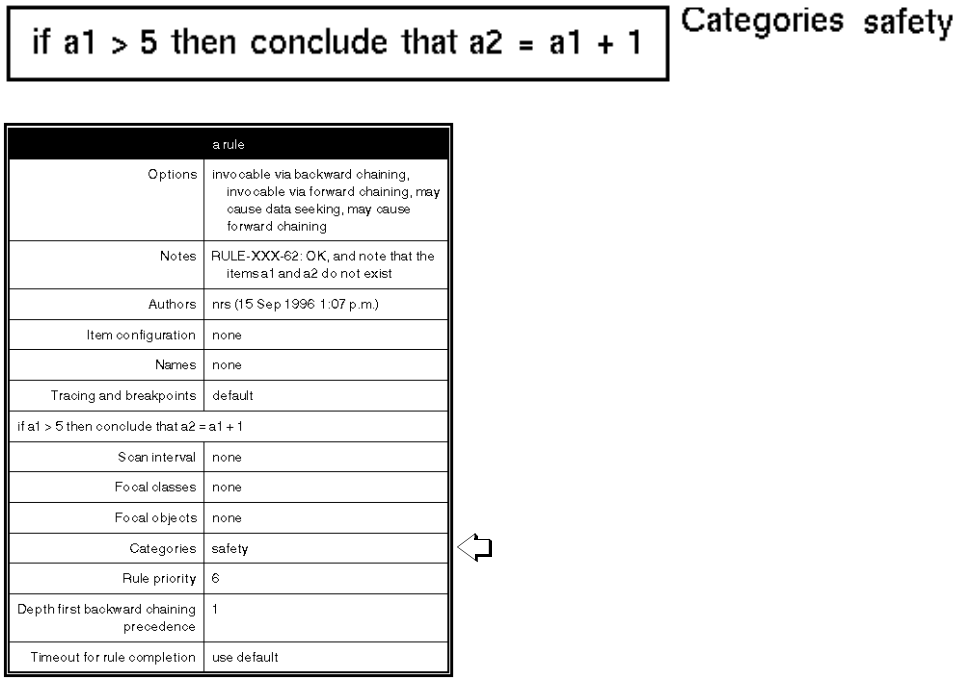
Specifying Which Categories of Rules the Rule Terminal Invokes

You can specify that the rule terminal invoke only certain categories of rules using the Rule Category attribute. You use this attribute in conjunction with the Categories attribute defined by a rule to invoke only those rules whose Categories attribute contains the specified Rule Category. For a discussion of the Categories attribute for a rule, see the *G2 Reference Manual*.

You can use the Rule Category attribute for rule terminals on a workspace or on a Single Source Encapsulation block.

Note In general, you should only specify the Rule Category attribute for invocation rule terminals. If you specify the same Rule Category attribute for both an invocation and conclusion rule terminal, an infinite loop will occur. For information on when you would specify a Rule Category for a conclusion rule terminal, see "Specifying a Rule Category for a Conclusion Rule Terminal" on page 609.

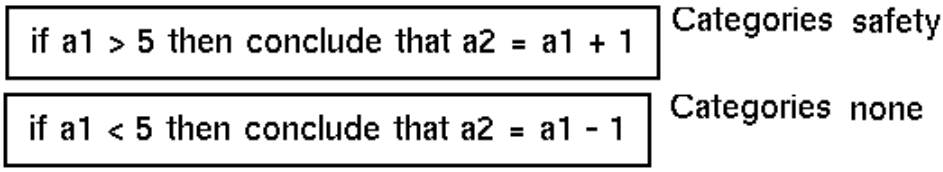
For example, suppose you have a set of rules that test the safety of an operation. In the table for the rule, specify the Categories attribute as safety. The following figure shows a rule with its Categories attribute displayed and its associated table:



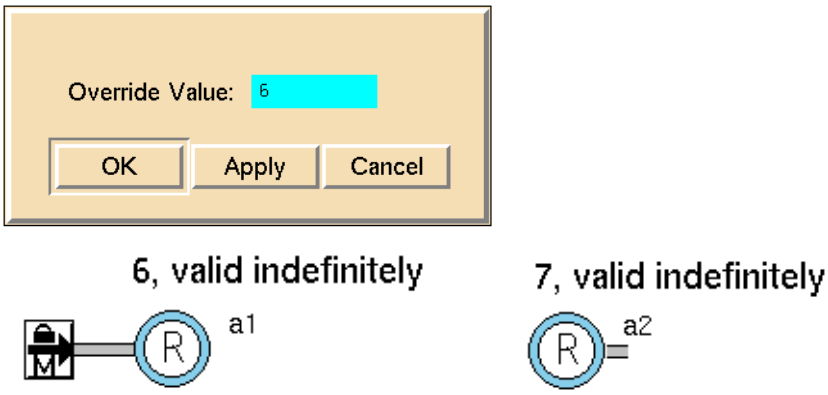
Note You can specify a list of category names for the Categories attribute. The symbols should be separated by commas and spaces.

To invoke only the safety rules using the rule terminal, specify Rule Category as safety in the configuration panel for the rule terminal.

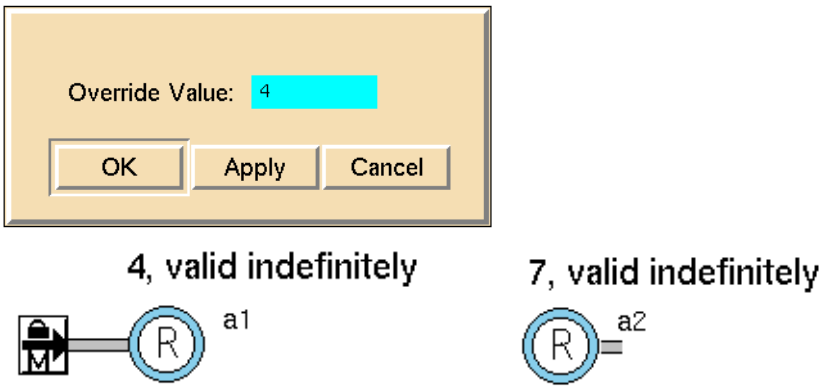
For example, suppose you have two rules, only one of which is in the safety category:



In the following example, the rule terminal A1 specifies Rule Category as safety. Thus, a value for A1 greater than five invokes the safety rule, which concludes a new value for A2. The rule terminals show the Last Recorded Value attribute values.



A value of less than five, however, has no effect on A2 since the safety rule cannot conclude a new value for A2:



Note If the rule specifies a value for Categories, and the rule terminal specifies Rule Category as none, the rule is invoked. In other words, a Rule Category of none invokes all rules.

Specifying a Rule Category for a Conclusion Rule Terminal

As stated above, you generally only specify a Rule Category for invocation rule terminals. You can, however, specify a Rule Category for a conclusion rule terminal, which is different from the Rule Category for an invocation rule terminal.

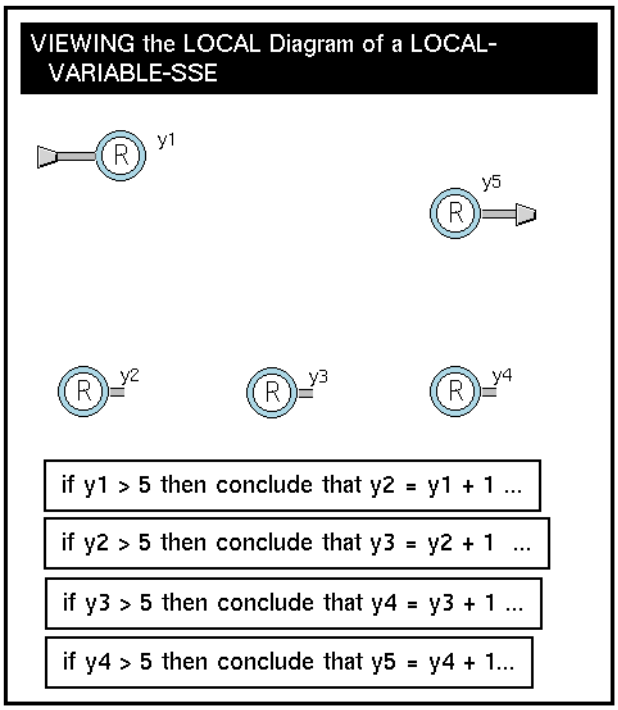
When the conclusion rule terminal concludes a value, the rules in its associated Rule Category are invoked. In this way, you can use conclusion rule terminals to invoke a different set of rules from invocation rule terminals.

Storing Local Values on an SSE Block Using Conclusion Rule Terminals

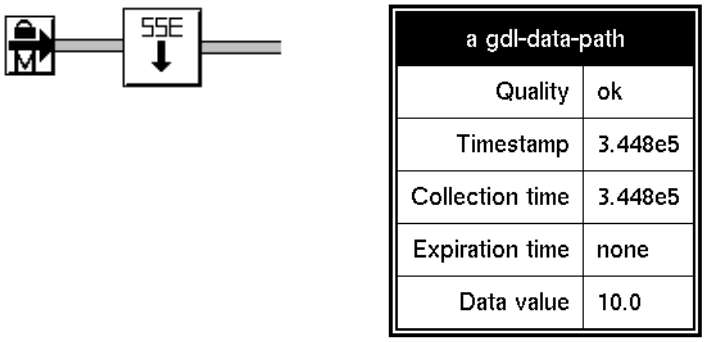
You use conclusion rule terminals that are not connected to any object to store local values on each instance of a Local Diagram of a Single Source Encapsulation block. These conclusion rule terminals both invoke rules and conclude values.

For example, suppose you have several rules on the Local Diagram of an SSE block, each of which tests the value of one rule terminal and concludes a value for another rule terminal, in sequence. The entry connection post is connected to an invocation rule terminal, and the exit connection post is connected to a conclusion rule terminal. The intermediate rule terminals are conclusion rule terminals that store local values on the Local Diagram.

In the following figure, y1 is an invocation rule terminal, y5 is a conclusion rule terminal, and y2, y3, and y4 are all conclusion rule terminal that store local values on the SSE block:



The following figure shows the result of entering a value of 6 into the SSE block:



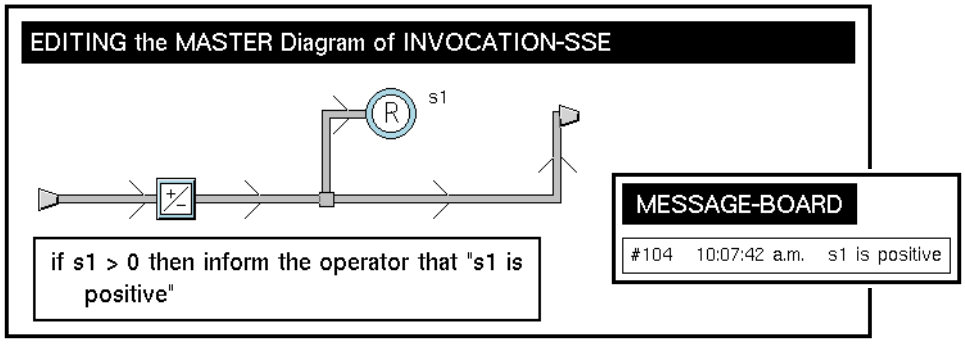
Note You can also connect intermediate rule terminals used as local variables to other GDA blocks for additional processing on the SSE.

Using Invocation Rule Terminals Alone on an SSE Diagram

You use invocation rule terminals alone on an SSE diagram when the antecedent of the rule specifies the name of the rule terminal, and the consequent of the rule performs some other action.

For example, suppose you have a rule terminal that invokes a rule on an SSE whose consequent sends a message to the operator if the value of the rule terminal is above a threshold.

In following figure, `s1` is an invocation rule terminal that evaluates when the block receives a value of `-3`. The SSE block changes the sign of its input and informs the operator when the value is positive.



Specifying the History of Values to Maintain

Rule terminals maintain a history of values, which can expire.

To determine the time at which data expires, specify the attribute Validity Interval.

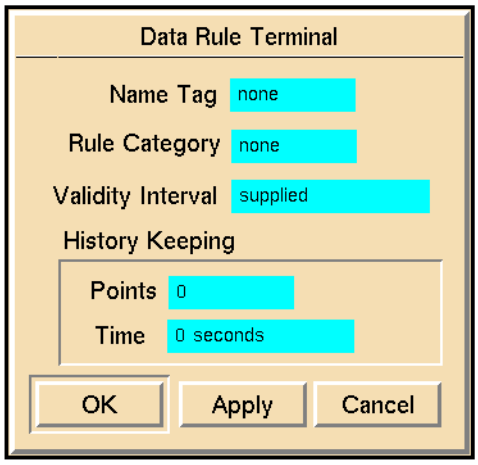
To determine whether the history is maintained using points or time, specify the attributes History Keeping Points or History Keeping Time.

Note You should specify History Keeping Points or History Keeping Time, but not both. If you do specify values for both attributes, GDA uses the history specification that results in the fewest number of sample points.

For general information about history attributes, see "Maintaining a History of Values" on page 84 in the *GDA User's Guide*.

Configuring

This is the configuration panel for a Data Rule Terminal. The panel for the other types of rule terminals are identical except for the name.



Attribute	Description
Name Tag	The name of the rule terminal to which a rule can refer.
Rule Category	A symbol that matches the specification of the Categories attribute of a rule, which limits the scope of the rules that the rule terminal executes.
Validity Interval	The amount of time that the history of values that the rule terminal maintains remains valid.

Attribute	Description
History Keeping Points	The number of points that the rule terminal should keep in its history.
History Keeping Time	The number of seconds for which the rule terminal should keep a history.

See Also

For more information on this block, see the sections below.

For more information on...	See...	In this book...
Editing Attribute Displays	page 27	*User's Guide*

27

Stub Tools

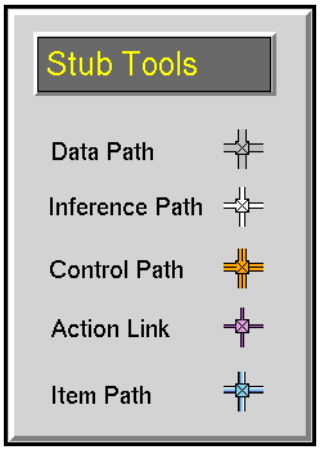
Describes the blocks that let you add connection paths interactively to custom blocks and encapsulation blocks.

Introduction

Stub tools allow you to add connection paths interactively to these types of blocks:

- Custom blocks
- Encapsulation blocks
- Single Source Encapsulation blocks
- Peer input blocks

You find the Stub Tools palette under the Other submenu of the Palettes menu:



You can create five types of stubs, in two directions, depending on the type of data: Data Path, Inference Path, Control Path, Action Link, and Item Path stubs.

For general information on data, inference, and control paths, see “Using Paths” on page 11. For general information on action links, see “Using Links” on page 66 in the *GDA User’s Guide*.

Data Path, Inference Path, Control Path, Item Path, Action Link Stub Tools

Stub tools allow you to create connection stubs for custom blocks, encapsulation blocks, single-source encapsulation blocks, and peer input blocks.

You use stub tools to create stubs in two places:

- In the Custom Class Wizard when you create stubs for subclasses of GDA custom blocks.
- On a workspace when you create stubs for Encapsulation blocks and Peer Input blocks (input stubs only).

For information on how to create stubs when you create custom subclasses of GDA blocks, using the custom block wizard, see "Customizing the Connection Stubs" on page 175 in the *GDA User's Guide*.

Note You can also create stubs by dragging any connection path from an existing block into the new block on a workspace, and then deleting the connection between the blocks. However, you can only use this approach when you create stubs on Peer Input blocks (input stubs only) and Encapsulation blocks.

Creating the Connection

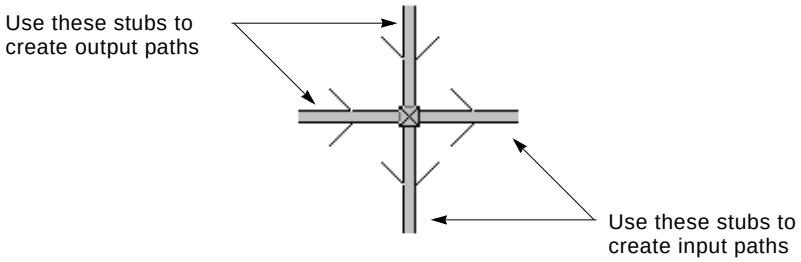
To create a connection stub for a block:

1. Clone a Stub Tool from the palette.
2. Place it next to a block.
3. Drag one of the four stubs into the middle of the block.
4. Click to connect the stub.

GDA automatically disconnects the Stub Tool, leaving the stub connected to the block.

Be sure to connect the stubs so that the direction of flow of the data is correct. To do this, always connect input stubs using the right or bottom stub, and always connect output stubs using the left or top stub.

The following diagram shows the direction of the stubs on a Data Path stub tool:



Naming the Ports

When you use the Stubs Tools to create stubs on custom blocks, using the Custom Class Wizard, GDA prompts you for the port name. GDA supplies a default port name, depending on whether it is an input or output port, and depending on the path type.

You can use any valid symbol when naming ports; do not choose any of the reserved words in G2, for example, in and out.

You can use descriptive names that indicate the type of the value being passed, such as Temperature, or you can use standard GDA portnames, such as dp-in and dp-out.

Be sure that each port name for a particular block is unique for the particular block.

In general, you should always name the ports of custom blocks. The exception to this is a Peer Input custom block, which does not require input port names. When you create input ports for a custom peer input block, using the Custom Class Wizard, the default port name is none, which automatically creates an unnamed port.

Using the Various Types of Stubs

There are five types of stubs that you can create for a block:

This type of stub...	Performs this task...
Data Path	Passes numeric, text, or symbolic values
Inference Path	Passes belief values (a number between 0 and 1) and status values (.true, .false, or unknown)
Control Path	Carries control signals

This type of stub...	Performs this task...
Action Link	Serves as a pointer to a target block
Item Path	Passes item instances

You can use any of the five types of stubs to create an input or output port for any block. The one exception is an Action Link, which you can only use to create an output port for a block.

Typically, you create data paths, inference paths, action links, and item paths on either the left or the right of a block, and you create control paths on either the top or the bottom of a block.

Configuring

The Stub Tools have no configurable attributes.

Examples

For examples of using stub tools to create Encapsulation blocks and Single Source Encapsulation blocks, see "Creating the Stubs for the Encapsulation Block" on page 479.

For examples of using stub tools to create Custom Blocks, see "Customizing the Connection Stubs" on page 175 in the *GDA User's Guide*.

For examples of creating item path connection stubs used with custom blocks, see the *GDA API Reference Manual*.

Glossary

A

action block: Performs actions on other blocks or on the environment when an inference value becomes true.

action link: Serves as a pointer when a block is to perform an action on a target object. For example, you can reset a block by using a control signal by attaching any block to the action link associated with the Reset block. When the block receives a control signal, GDA resets the block. Action links appear on many blocks on the Action menu.

alarm: Indicates the source of potential problems in the real-time data that the diagram is monitoring. GDA displays alarms in the Alarm Queue.

API procedure: When you are creating custom subclasses of custom blocks, statements in the block evaluator procedure call Application Programmers Interface (API) procedures that:

- Set values onto a path and trigger the evaluation of downstream blocks.
- Set values onto a path without triggering the evaluation of downstream blocks.
- Obtain values from a path.
- Resolve output path attributes for custom blocks.

attributes: Specify the particular behavior of a block. You specify block attributes in the configuration panel.

B

block evaluator: Executes when a custom block receives a value on its input path(s). GDA supplies a template for the block evaluator, based on the input and output stubs and the type of block. You specify in the custom portion of the procedure.

block menu: Enables you to perform G2 operations on blocks, such as cloning, transferring, deleting, and configuring.

C

capability link: Adds various types of features to blocks, such as charts, graphs, and clocks.

configuration panel: A dialog that enables you to specify the attributes of a block.

configure: To specify the attributes of a block in a configuration panel or in the attribute display of a block.

connection post: Enables a block on one workspace to pass data to a block on another workspace.

control path: A type of path that passes control signals, which cause downstream blocks to execute.

custom block: A subclass of GDA block that executes custom G2 procedures. There are three basic types of custom blocks:

- General - performs calculations on single or multiple inputs when the block needs to reference specific inputs by port name.
- Peer Input - performs calculations on single or multiple inputs when the block treats all inputs equally.
- Multiple Invocations - enables control over how the block processes simultaneous control signals.

Custom Class Wizard: Creates new subclasses of GDA blocks and edits existing subclasses. Each instance of a custom subclass inherits the definition of an existing GDA custom class or a custom block subclass.

D

data block: Operates on numeric, textual, or symbolic values.

data path: An input or output path to a block that contains numeric, textual, or symbolic values.

description: An attribute of certain inference blocks that enables you to describe the current output value and provide advice to operators. For example, if the inference block passes .true, the description might be the temperature is too high.

developer mode: A user mode that provides access to all the basic functionality required for building schematic diagrams.

disable evaluation: Stops a block from passing its output value and responding to new values. Disabling evaluation enables you to "turn off" entire portions of a GDA diagram.

discrete logic: A form of inferencing whereby a block passes discrete inference values, for example, a Status-value of .true, .false, or unknown, and a Belief-value of 1.0, 0.0, or 0.5. *See also* fuzzy logic.

E

enable data input: To run a diagram, you must enable data input, which:

- Starts data flowing into entry points.
- Evaluates signal generators.
- Evaluates clock capabilities.

enable evaluation: Allows a block to pass its output value and respond to new values.

encapsulation block: Enables you to hide complexity in a GDA diagram by placing a portion of the diagram on a subworkspace. There are two types:

- Single Source Encapsulation blocks enable you to create multiple instances of the encapsulation block when you need to propagate changes in the definition of the encapsulation to other blocks in the diagram.
- Simple Encapsulation blocks are the same except that you cannot propagate the changes in the definition to other blocks in the diagram.

entry point: Receives data externally from a variable, a GSI (G2 Standard Interface) variable, G2 procedure, or from an embedded variable in the table for the block. Entry points are the starting point of a GDA diagram. GDA supports five kinds of entry points: numeric, text, symbolic, belief, and control.

evaluate: To execute the procedure for a block. For example, when you evaluate an entry point, the current value is propagated with a new timestamp. When you evaluate a block with a single input control path, the block acts as if it has received a new control signal.

explanation: Provides a description of the combination of upstream blocks that resulted in the current output inference value of certain blocks.

F

filter: A category of block that you use after data entry blocks in a diagram to filter out noise and find trends in data. GDA supports six kinds of filters: changeband, outlier, first-order exponential, nonlinear exponential, quadratic, and cubic.

fuzzy logic: A form of inferencing whereby a block passes a range of values, for example, a Status-value of .true, .false, or unknown, and a Belief-value that is a number between 0.0 and 1.0, where 0.0 is completely false and 1.0 is completely true.

G

G2 Main Menu: Controls whether G2 is running or paused, and allows you to load and save applications.

G2 menus: Provide all the functionality of G2 within the GDA environment, for example, the creation of class definitions, variables, parameters, rules, and procedures.

H

history: A store of past input values. Numerous GDA blocks operate on the stored values, such as computing the average of the last 25 input values.

HTML: Hypertext Markup Language. Online documentation is a collection of HTML files, which you can display in any HTML browser.

hysteresis: Passes the previous value in a condition block if the current changes are small. For example, if the belief value of a block whose Output Uncertainty is 0.5 changes from 0.8 (.true) to 0.7 (unknown), the block will continue to pass .true if the Hysteresis When attribute is set to .true.

I

inference block: Translates data values to truth values and operates on truth values, including fuzzy truth values. For example, observations observe data values and pass inference values, and logic gates use Boolean logic to combine reference values.

inference path: A type of path that carries truth values. Inference paths carry two values:

- Belief value - a number between 0.0 to 1.0, where 0.0 iscompletely false and 1. 0 iscompletely true.
- Status value - one of the symbols .true, .false, or unknown. GDA derives status values from belief values.

initial value: The value a block passes when you first start G2 or when you reset the block.

input port: Carries data to a block. A block has one or more input ports depending on the type of block.

invoke: To execute the procedure for a block. GDA invokes a block when:

- An entry point receives a value from its data source.
- A value is propagated onto the input path of the block due to the behavior of an upstream block.
- You evaluate a block manually.

item path: A type of path that passes any G2 item through the block. Use item paths with custom blocks for:

- Discrete event processing, for example, processing individual items in an assembly line.
- Complex data processing, for example, processing multiple items, using an item list or item array.

K

KB Workspace menu: Enables you to create G2 definitions and objects on a workspace and set up applications.

L

link: A special-purpose type of connection that you use to add features or behaviors to a block, or to perform actions on a block. For example, you use links to add a graph or chart capability to a block.

There are three types of links:

- Action
- Capability
- Restriction

lock: When locked, a block does not respond to input data or pass its output values. GDA locks a block when you manually override its value.

M

message: GDA sends messages to the Message Queue when you use the Send Message block.

multiple values: Simultaneous control signals that a block receives. A block can ignore or use multiple values as needed.

N

no-value inputs: Occur when stubs are unattached or when connected paths never receive a value. Peer input data blocks and peer input logic blocks ignore input paths with a Quality of no-value. Non-peer input blocks require all of their inputs to evaluate, and, therefore, never place a value onto an output path if the block has a no-value input.

O

observation: A category of blocks that detect features in your data. Observation blocks take data as input, test it against a threshold, and pass as output the inference value that the test produced.

output port: Carries data from a block. A block has one or more output ports depending on the type of block.

override: To manually change a block's output value for testing purposes. Overriding a block locks the block and propagates a Quality of manual onto the output path.

P

parameter: A G2 object that stores a data value and keeps a history of it over a specified time. A parameter can also initiate forward chaining.

path: The connection between two blocks. GDA supports data, inference, control, and item paths.

path attributes: Attributes that provide information about the value and status of the data on one path. Each type of path defines slightly different path attributes.

path quality: A path attribute that specifies the status of a path's data. There are three types:

- Manual
- No-value
- Expired

path splitter: Connects the input stub from one block to the path between two other blocks so that more than one downstream block can get input from the same upstream block.

peer input block: A category of block that can have any number of inputs and does not evaluate the inputs in a specific order. The inputs are treated all alike and are therefore peers.

port: Connection stub to which another stub can attach. Ports are either named or unnamed, depending on the type of block and whether the port is input or output.

Q

queue: A special workspace that displays information about alarms, explanations, messages, and errors.

R

reset: Causes the following to happen:

- Sets the block to its initial state, which propagates the block's initial value.
- Clears any error conditions.
- Unlocks the block, if it was locked.
- Erases the blocks history if the block maintains a history.

restriction link: Enables you to customize a block's capabilities. You can customize the override dialog and determine the source of input for a GDA diagram.

S

signal generator: Generates a continuous signal to a diagram, to simulate real-time data. Examples of signal generators include the Sine Wave signal and White Noise signal.

snapshot: A file that contains the current state of the running application as backup. You can configure GDA to take snapshots automatically at regular intervals. When you restore a snapshot, GDA resumes running the application from the point at which you took the snapshot.

Statistical Process Control (SPC): A category of block that uses statistical methods to measure the quality and consistency of a process. The blocks in the SPC palette contain statistical tests, pattern recognition techniques, and graphs. An application that uses SPC monitors a process and compares current performance with expected performance.

stub: A connection port to which another connection port can attach.

sweep: An internal mechanism where GDA searches for invoked blocks, evaluates them, and continues until no more blocks are left to evaluate.

system administrator: A category of GDA users who works in Administrator mode, which enables access to additional attributes and menu choices used for debugging.

T

temporal logic: A type of inferencing that allows you to analyze the timing of events.

top menu bar: Provides access to basic functionality, including controlling the diagram, cloning blocks from palettes, accessing queues, and customizing the environment.

U

uncertainty: Defines a band around 0.5 that determines the status value unknown. For example, if the attribute Output Uncertainty is 0.25, then the Belief-value is . true above 0.625 and .false below 0.375 and Unknown between .65 and .35.

user modes: There are four user modes:

- Administrator - enables system administrators to access attributes and menu choices used for debugging.
- Developer - enables developers to access all the basic functionality used for building schematic diagrams.
- User and Browser - enables end users to view diagrams, display menus, and display configuration panels of objects but not to move blocks, clone blocks, or edit attributes. Browser mode is slightly more restrictive than User mode.

V

variable: You use variables as starting points in a GDA diagram, either by referring to the variable in an entry point, or by connecting blocks directly to the variable. *See also* parameter.

vertex: An 90° bend in the connection between blocks.

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z